On Improving the Performance of the Gauss-Newton Filter

Roaldje Nadjiasngar

A thesis submitted to the Department of Electrical Engineering,
University of Cape Town, in fulfilment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering.

Cape Town, November 2013

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of PhD in Engineering in the University of Cape Town. It has not been submitted before for any degree or examination in any other university.

Signature of Author .....

Cape Town

November 2013

Abstract

The Gauss-Newton filter is a tracking filter developed by Norman Morrison around the same time as the celebrated Kalman filter. It received little attention, primarily due to the computation requirements at the time. Today computers have vast processing capacity and computation is no-longer an issue. The filter finite memory length is identified as the key element in the Gauss-Newton filter adaptability and robustness. This thesis focuses on improving the performance of the Gauss-Newton. We incorporate the process noise statistics into the filter algorithm to obtain a filter which explains the error covariance inconsistency of the Kalaman filter. In addition, a biased version of the linear Gauss-Newton filter, with lower mean squared error than the unbiased filter, is proposed. Furthermore the Gauss-Newton filter is adapted using the Levenberg Marquardt method for improved convergence. In order to improve the computation requirements, a recursive version of the filter is obtained. The recursive version of the filter has an exponential forgetting factor, a substitute of the non recursive filter memory length. It is shown that the recursive Gauss-Newton filter is equivalent to the iterated extended Kalman filter when the forgetting factor is equal to one. Stability analysis has demonstrated the necessity of having finite memory

length, thus imposing a less than unity forgetting factor. The forgetting factor is updated by means of a memory control algorithm, similar to Morrison's master control algorithm (MCA). The new compact filter retains the robustness and adaptability of the non recursive filter, and stands out as a good candidate for tracking highly manoeuvrable targets. The recursive filter is also adapted using the Levenberg Marquardt method.

To my dad late Nadjiasngar Neddje."I got no other inheritance for you but education", these words said to me a few months before he departed have been my companions ever since.

In memory of my father

Nadjiasngar Neddje

1946–1997,

and my mother

Basmath Assane

1948–2012

Acknowledgements

I would like to thank Professor Michael Inggs for his guidance during the research project. He has been an unfailing support, ever since the day I joined the Radar Remote Sensing Group (RRSG). My gratitude goes to Dr Morrison for introducing me to filter engineering and, in particular to the Gauss-Newton filter, which is the focus of this thesis. My Thanks go as well to Dr Yoann Paichard and all the members of the RRSG for making the environment ideal for this endeavour. Finally I would like to address my gratitude to KACST, and UCT for the financial support.

Contents

List of Figures

List of Tables

List of Symbols

λ	—	Forgetting factor
λ_w	—	Forgetting factor used for computation of RSSR
μ	—	damping factor
γ	—	gain ratio
L	—	The Memory length
$\mathcal{O}$	—	big o computational complexity
ξ	—	used for computation of the perturbation vector
$M(X)$	—	observation sensitivity matrix evaluated at X
$A(X)$	—	State sensitivity matrix evaluated at X
$\mathbf{T}$	—	Total transition matrix
$\mathbf{W}$	—	Filter matrix for exactness constraint
W^{-1}	—	Filter covariance matrix

X_n	—	the state vector
Y_n	—	the observation vector
R_n	—	observation covariance matrix
δX_n	—	perturbation vector
δY_n	—	perturbation observation vector
$\mathbf{Y}_n$	—	the total observation vector
$\mathbf{R}_n$	—	the total observation covariance matrix
$\delta \mathbf{Y}_n$	—	the total perturbation observation vector
B	—	Bias matrix in the bias estimator
Q	—	process noise covariance matrix
$\varepsilon(E)$	—	mean of random variable E
$E_n(\delta X_n)$	—	cost function as a function of δX_n
σ	—	Covariance matrix shrinking parameter
S_σ	—	The shrunk covariance matrix

Nomenclature

RMSE—Root mean squared error.

MSE— Mean Squared error.

GNF—Gauss-Newton Filter.

RGNF—Recursive GNF.

LM— Levenberg Marquardt

LMA— Levenberg Marquardt Algorithm.

IEKF—Iterative Extended Kalman Filter.

MCA—Master Control Algorithm.

ASCA—Adaptive Switching and Control Algorithm.

DPV— Differential Perturbation Vector.

SSR—Sum of squared residuals.

NSSR—Normalized sum of squared residuals.

RSSR—Recursive sum of squared residuals.

RSSR—Normalized recursive sum of squared residuals.

Chapter 1

Introduction

The Gauss-Newton filter is an unbiased estimator based on the minimum variance theory. The filter is equivalent to the maximum likelihood estimator and has a Cramer-Rao lower Bound.

It has been over 40 years since this estimator was introduced, and yet it has received little attention, primarily due to the higher computational requirements when compared to the Kalman filter. Today computers have vast processing capacity and are increasing exponentially into the future. It is therefore safe to argue that computational load alone should not be considered when selecting an estimator.

Table1.1 highlights the majors attributes of the GN filter in comparison to the Kalman filter.

1 The Gauss-Newton can estimate from stage-wise correlated data. Correlation occurs when data is observed by failing sensors or from different types

Key Attributes	GNF	Kalman
Correlated data	yes	no
Data updating time	any time stamp	constant interval
Suitability to highly changing dynamics	yes	no
Initialization	no	yes
Tracking process with driving noise	yes	yes
Recursive computation	no	yes

Table 1.1: GNF key attributes

of sensors.

2 The filter measurement update can be done at any time stamp. This particularity allows long term prediction.

3 The filter possesses a finite memory length that can be varied in an adaptive manner for estimation of highly changing dynamics. The memory length acts like a sliding window and therefore allows the filter to focus on the most recent observation, hence minimising the effect of previous dynamics on current estimation.

4 The GN was derived by assuming deterministic dynamics and this was believed to be the limitation as many systems have additive process noise. However it is shown in this thesis that the GN filter can easily be adapted for processes with dynamic noise. In fact the finite memory nature of the filter provides the best means of mitigating the effect of the process noise covariance matrix which requires fine tuning in the Kalman filter [1]. The adapted filter explains the error covariance inconsistency of the Kalman filter [2, 3, 4].

5 It is well documented that the Kalman filter requires good initialization

for convergence. The GN on the other hand does not require initialization since the covariance matrix is computed in accordance with the available batch of data. In the non-linear estimation problem an initial guess of the process state is required to start the iteration. However this can be obtained from the available observation.

6 The GN filter is iterative and non recursive. The processing time is dependent on the size of the filter memory.

The derivation of the recursive form of the GN has demonstrated that the Kalman filter is equivalent to the GN filter when the noise in the data is uncorrelated, data is sampled at a constant rate and the filter memory is continuously expanding. The later characteristics explains the Kalman filter's inability to handle rapidly changing dynamics.

The Recursive form of the GN filter has a forgetting factor that acts as the substitute to the filter memory, hence it can be changed adaptively to estimate a manoeuvring target.

In Figure 1.1 we show in grand scheme the relationship between the GNF and the Bayesian estimator, considered to be the most optimum estimator. Due to the equivalence in their formulations, the estimators such as Swerling filter and Kalaman and Bucy filter are all put in one category , that is the Kalman filter [3, 4]. The Baye's filter or the Bayesian estimator assumes the noise in the observation is stage-wise uncorrelated. However it is simply a recursive formulation of the minimum variance with no restriction placed on the probability density function [4]. For a Gaussian estimation problem the Gauss-Newton fil-

ter would be the optimum estimator when the noise is stage-wise correlated. The recursive Gauss-Newton would be equivalent to the Kalman filter when the forgetting factor is unity.

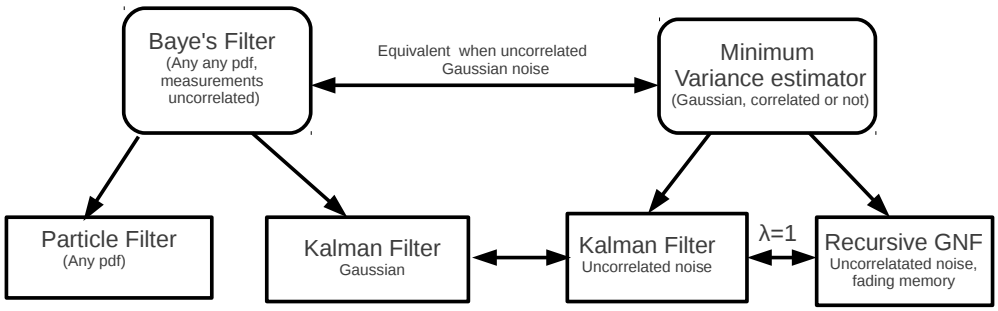

Figure 1.1: Relationship between Bayesian estimator and the GNF

1.1 Problem Formulation

The GNF algorithm, although robust, requires significant processing power, i.e. the amount of memory required. To improve the computational efficiency of the GNF, studies of the use of Field Programmable Gate Arrays (FPGA) and other co-processor technology have been made [5, 6]. Memory requirements were iden-

tified in these studies as being the major stumbling block in implementations both on FPGA (low power and parallelism) and coprocessor (ease of use) technology. Therefore, we found it is necessary to investigate alternative methods of improving the filter algorithm in order to reduce the computation burden. Such methods, if existent should not alter the key attributes of the filter - robustness and flexibility.

Another problem which may occur is the singularity of the Hessian matrix. A singular Hessian will undoubtedly degrade the filter performance.

This thesis proposes a derivation of the recursive form of the GNF that requires zero memory. It is shown that the iterated extended Kalman filter (IEKF) is a limited version of the RGNF. The filter possesses an exponential forgetting factor, an equivalent to the GNF memory length that provides the filter with a higher degree of flexibility.

To address the problem of singularity, both the GNF and the RGNF are adapted to the Levenberg-Marquardt method to obtain more robust type of filters [7].

1.2 Publications

The following papers are published as results of the thesis research:

1 R. Nadjiasngar and M. Inggs, Gauss-Newton Filtering incorporating Levenberg marquardt methods for tracking, Digital Signal Processing, no. 0 pp. 2013.

2 R. Nadjiasngar, S. Middleton, and M. Inggs, Doppler-only tracking with the recursive Gauss-Newton Filter, in Radar Systems (Radar 2012), IET International Conference on, pp. 1-5, 2012.

3 R. Nadjiasngar, M. Inggs, Y. Paichard, and N. Morrison, A new probabilistic data association filter based on composite expanding and fading memory polynomial filters, in Radar Conference (RADAR), 2011 IEEE, pp. 152-156, may 2011.

1.3 Overview of the Thesis

1.3.1 Gauss-Newton Filter

Chapter 2 describes the mathematical formulation of the Gauss-Newton filter. The relationship between the filter, the Cramer-Rao lower bound and the maximum likelihood estimate are demonstrated. Then follows the discussions on the filter memory length with emphasis on how the filter memory length can be adaptively varied for estimation in highly non-linear systems. The end of Chapter 2 explains the singularity of the Hessian matrix which may occur and hence may cause performance degradation. Chapter 2 also refers to Chapter 4 which presents solution to the singularity problem.

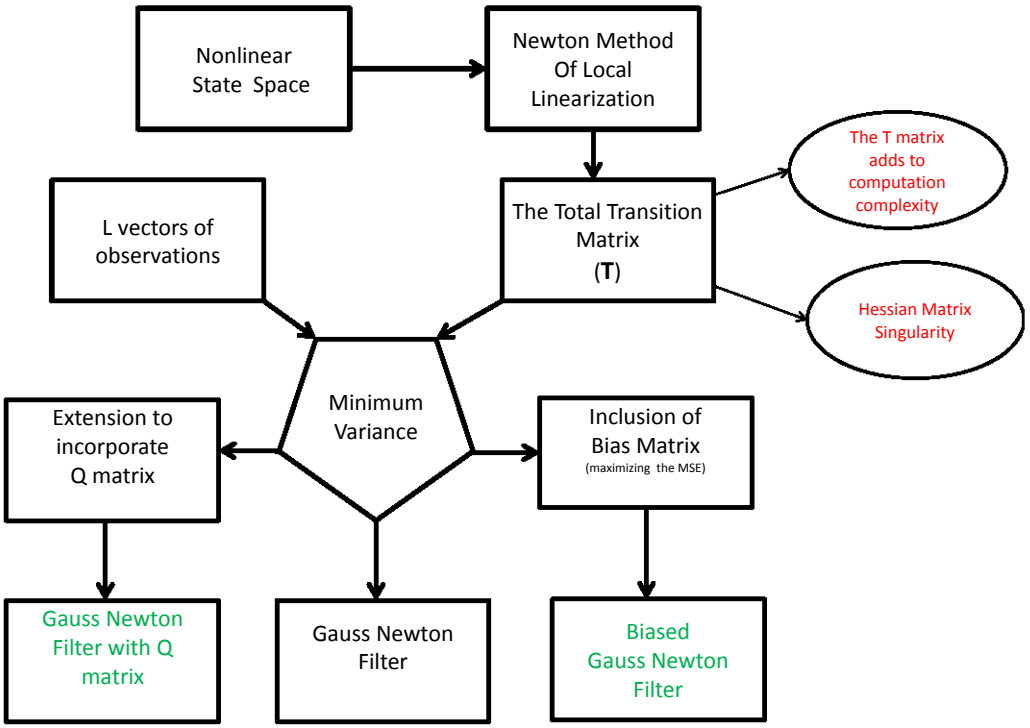

Figure 1.2: Overview of the the GNF

The minimum variance estimation

The minimum variance algorithm has been used to estimate parameters from batches of observations, accumulated over a defined period of time. The most popular version of the minimum variance methods is the weighted least squares method, which is at the heart of adaptive filtering [8] [9]. For the estimation of non-linear state space models, a non-recursive filter called the Gauss-Newton filter (GNF) was developed and has been successfully used in many applications [3] [10] . The GNF algorithm is a combination of the Newton method of local linearisation and the least squares-like version of the minimum variance method

(Figure 1.2. It is used to estimate process states that are governed by non-linear, autonomous, differential equations, coupled with linear or non-linear observation schemes. The Filter algorithm consists of three main steps:

1 accumulation of the observations vectors, along with their respective covariance matrices, over a memory length L.

2 Construction of the total transition matrix $\mathbf{T}$ through back propagation of the predicted state over the memory length L.

3 The minimum variance is then applied for state estimation.

Being derived directly from the minimum variance the GNF has a Cramer-Rao lower bound which is its covariance matrix. Furthermore the GNF estimate is the same as the maximum likelihood estimate in a Gaussian environment.

The Filter memory length

The filter memory length is the parameter that provides the GNF with a high degree of flexibility and adaptability. The GNF always obtains the estimate that maximizes the posterior distribution. The accuracy of the estimate depends on the quantity of the data available. When L is high, the filter assumes the dynamic model is the same over the L sized window and therefore it focuses on obtaining much smoother data. On the other hand, for highly changing (dynamic) models a shorter memory length allows the filter to focus on the most recent behavior of the systems. Furthermore, at low L the filter is at initial state and the dynamic

model can be changed smoothly. The master control algorithm described in [3] is an efficient and adaptive method for selecting L.

Process with driving noise

The derivation of the Gauss-Newton filter assumes a deterministic dynamic and it was believed that this put some limitation on the filter's ability to estimate process with driving noise. However, it can be shown that the minimum variance can be extended to incorporate systems with process noise. The estimator with process noise is the same as the derived deterministic. The only difference is that the weight matrix includes the process covariance matrix Q. Simulation studies have shown that Q matrix is not crucial to the GNF filter. The effect of the process noise can be mitigated by only reducing the filter memory. It was observed that having the Q matrix improves the estimation when $L < 50$, but it creates error covariance inconsistencies for bigger memory lengths. From these findings, it is evident that the memory length of the filter is the best tool for mitigating the effect of the process noise and at the same time maintaining error covariance consistency.

These observations have motivated further investigation into the role of the Q matrix. We finally establish mathematically that the Q matrix reduces mean squared error but increases the variance of the estimate, hence it creates error covariance inconsistency.

Task	Final Size	Complexity
$\mathbf{R}^{-1}$	$mL \times mL$	$\mathcal{O}(Lm^2)$
$\mathbf{T}$	$mL \times k$	$\mathcal{O}(Ln^2 + Lmk^2)$
$\mathbf{TR}^{-1}\mathbf{T}$	$n \times k$	$\mathcal{O}(Lmk^2 + km^2L^2)$
$(\mathbf{TR}^{-1}\mathbf{T})^{-1}$	$k \times k$	$\mathcal{O}(k^3)$
$\mathbf{TR}^{-1}\mathbf{Y}$	$k \times 1$	$\mathcal{O}(km^2L^2 + Lmk)$
$\hat{\mathbf{X}}$	$k \times 1$	$\mathcal{O}(k^2)$

Table 1.2: Computational complexity

The Bias Estimator

Considering the linear form of the Gauss-Newtton filter, a bias filter with superior mean squared error can be obtained. The approach in [11] is used to derive such a filter. Details on the new filter equation can be found in Chapter 2.

Computational Complexity

The memory nature of GNF makes it computationally intensive. Table 2.1 shows the computational complexity involved in each stage of the filter operation. The major contributors are the computation of the $\mathbf{T}$ matrix and the Hessian matrix $(\mathbf{T}_n^T\mathbf{R}_n^{-1}\mathbf{T}_n)$. For single cycle of the filter the required computation is of the order $\mathcal{O}(km^2L^2)$, where k is the size of the state vector and m the size of the measurement vector. This result is obtained assuming $L \gg k$. Therefore the GN computation requirements increase quadratically with the memory length. The computation load of the GNF is very significant when compared to the Kalman filter or the recursive GNF which has complexity of $\mathcal{O}(k^3)$ or lower (when matrix inversion is avoided).

1.3.2 The recursive Gauss-Newton Filter

While The GNF memory plays a key role in the filter's robustness and flexibility, it is also the main source of the intensive computation requirement. To address the memory issue, Chapter 3 presents the derivation of a set of recursive equations of the GNF. Further expansion of the recursive equations has demonstrated the equivalence of the Gauss-Newton filter to the iterated extended Kalman filter (IEKF) when the filter memory is unbounded. Furthermore, the stability analysis demonstrates the necessity of having finite memory length. An adaptive forgetting factor algorithm, the adaptive switching and control algorithm (ASCA), based on the recursive sum of squared residuals is developed.

Stability of the filter

The basic form of the recursive GNF follows the discrete Lyapunov equation. The discrete Lyapunov equation includes a forgetting factor λ , a substitute to the filter memory length. The forgetting factor must be less than 1 for neutral stability. It is noted that the stability bound expands as λ decreases. This finding confirms the role of the memory length in the filter robustness.

Relationship with the Kalman filter

The derived recursive equation can be organized in many ways. When matrix inversion lemma is applied on the discrete Lyapunov equation it results in the Kalman filter covariance expression. Further manipulations have shown that the recursive GN filter is equivalent to the iterated extended Kalman filter when

$\lambda = 1$. That is, the Kalman filter is equivalent to the GNF when considering unbound memory length.

The Adaptive Switching and Control Algorithm (ASCA)

In Chapter 2 we arrived at a conclusion that the memory length of the GNF provides the filter with a high degree of flexibility. The RGNF filter has a forgetting factor which is related to the GNF memory length ($L \approx \frac{1}{1-\lambda}$). In this chapter we describe how the forgetting factor can be varied adaptively to track manoeuvres. We follow Morrison's approach on using the sum of squared residual(SSR) to adaptively vary the non-recursive GN memory length [3]. The SSR here is computed recursively. Figure 1.3 shows a block diagram describing the operation of the ASCA. The ASCA allows a timely switching of the the filter model to track manoeuvres. The effectiveness of the ASCA is demonstrated in a simulation study.

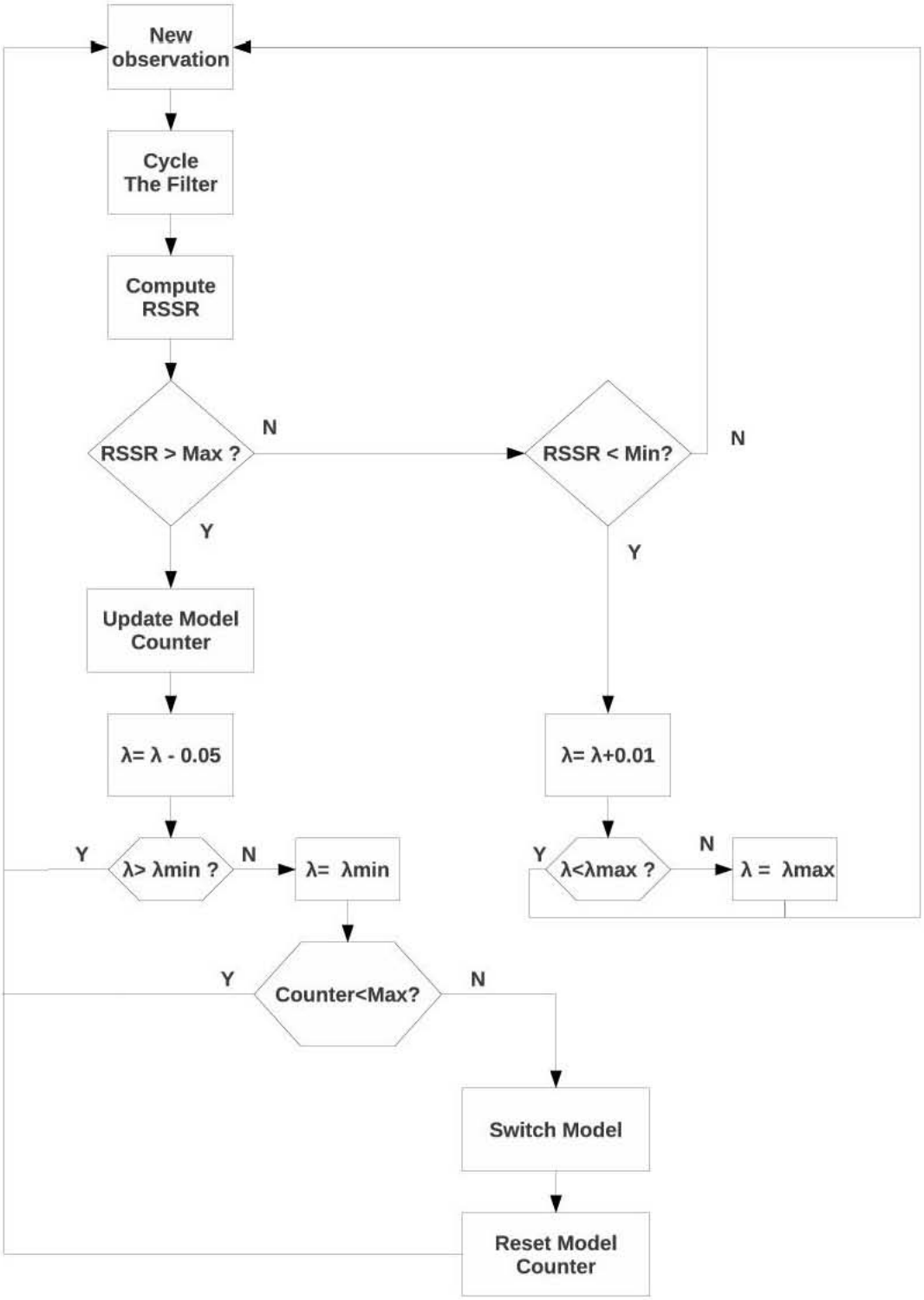

Figure 1.3: The adaptive switching and control algorithm based on the adaptive forgetting factor algorithm

1.3.3 The adaptation to Levenberg-Marquardt

Chapter 4 describes the adaptation of the GNF to the Levenberg Marquardt algorithms(LMA) [12] which results in a new tracking filter. The LMA are optimization algorithms well known for their powerful convergence properties. The adaptation of the GN filters to LMA requires a presence of an adaptive damping factor in the Hessian matrix that has a double role of improving the convergence of the filter as well as the elimination of the possible singularity of the matrix. The new filter can be used for radar targets tracking with improved convergence. To address the computation requirements a similar approach was used to adapt the recursive version of the filter to the LMA to obtain what we call LMA-RGNF. We also show through the simulation studies that the performance of both filters is not affected by the process noise whose knowledge is central to the family of Kalman filters.

GNF adapted to LMA

In Chapter 2 we show that the Hessan matrix of the GNF is the inverse of its covariance matrix. The condition for the uniqueness of the local minimizer is that the Hessian matrix must be positive definite. However this condition may not be met due to the nature of the observation scheme or due to numerical approximation. In addition, the GNF will fail to converge if the initial guess is far from the true state. To address these issues we follow the Levenberg Marquadt approach by introducing the damping parameter μ in the filter equation. When μ is large, the algorithm behaves as a steepest descent which is ideal when the

current solution is far from the local minimum. The convergence will be slow but is guaranteed. When μ is small, the algorithm has faster convergence and behaves like the Gauss-Newton. The damping μ is updated by the gain ratio which is computed from the cost function.

The RGNF adapted to LMA

For the RGNF W_n, the inverse of the filter's covariance matrix is replaced by $W_n + \mu I$ at each iteration. However the recursive equation of W_n remains unchanged. It is only during the iteration that the damping parameter is introduced. A comprehensive algorithm based on this adaptation is presented in great detail in Chapter 4. The newly adapted filter is self initializing with a faster convergence rate. It is noted that a lower value of forgetting factor increases the convergence rate of the filter. However, an excessive reduction of the forgetting factor will result in a noisy estimate. The memory control algorithm developed in Chapter 3 can effectively be used with the Levenberg Marquardt-RGNF (LM-RGNF).

1.3.4 Concluding remarks

The Gauss-Newton filter is a robust estimator. It is derived from the minimum variance unbias estimation theory and it has a Cramer-Rao lower bound which is its covariance matrix. In addition, the filter is equivalent to the maximum likelihood estimator in the Gaussian environment. All these properties have made the GNF very attractive. The main contributor to the filter's robustness is identified to be its memory length, which can be adaptively varied for the

estimation of systems with changing dynamics.

The earlier belief was that, by assuming a deterministic dynamic model, the filter cannot estimate a system's state with process noise. However we demonstrated experimentally that the filter's finite memory nature allows it to mitigate the effect of the process noise. To further understand the effect of the process noise on the estimation, we derived a version of the GNF which incorporates the process noise covariance matrix. It is noted that the only difference in the filter equation is the weight matrix, which now includes the process noise covariance matrix. Simulation studies have demonstrated that having the process noise covariance matrix only improves the filter estimation error for lower memory length. When L increases it was evident that the estimation error grows while the covariance matrix shrinks. This result is a typical case of error covariance inconsistency seen in the Kalman filter. We then established mathematically that the Gauss-Newton filter with process covariance matrix results in a lower mean squared error but, at the expense of a widened covariance matrix.

The major limitation of the GNF is the computation requirements which increases quadratically with respect to the memory. The relationship between the computational complexity can even be cubic when processing correlated data. To address the computation issue, we derived a recursive form of the GNF, the RGNF. The RGNF has a forgetting factor which is directly related to the GNF memory length L. Hence it can be adaptively changed, providing the filter with similar attributes as the GNF. An adaptive switching and control algorithm (ASCA) is developed to adaptively change the forgetting factor. The effectiveness of the algorithm is demonstrated in simulation and we think the ASCA

stands out as a good candidate for tracking maneuvering tracking.

Further studies of the RGNF have demonstrated that it is equivalent to the iterated extended Kalman filter when considering unbounded memory. The basic recursive equation that governs the inverse of the covariance matrix is the discrete Lyapunov equation, which plays an important role in the filter stability. The stability analysis has constrained the filter's forgetting factor value to less than 1. The implication of this result is that for neutral stability the filter should possess finite memory. These findings concur with the role of the memory in the robustness of the non recursive GNF.

We are aware of the GNF Hessian matrix which can lose its positive definiteness in some situations such as inappropriate observation scheme or numerical approximation. To remedy this problem we follow the Levenberg-Marquardt approach by introducing a damping parameter, which makes the filter a combination of Newton steepest descent and the Gauss-Newton method. The obtained filter is more robust than the GNF. It was noted that the singularity problem may also occur to the RGNF and the Kalman filter. Therefore the RGNF was also adapted to the Levenberg-Marquardt method.

Our conclusion is that the non recursive GNF remains the most robust type of filter. However we have decreased the gap significantly by obtaining the recursive form that has a forgetting factor. The presence of the forgetting factor allows estimation of time varying signal in the most optimum way. Further adaptation of the RGNF to the Levenberg Marquardt method results in a filter with self initialization capability. This is done at a slightly higher computation cost when compared to the models without a damping parameter.

Future studies would be to perform a complete analysis of the effect of the forgetting and damping factors (separately or combined) on data association. A deeper analysis of the Bias tracking filter and the Gauss-Newton filter with process noise is also required.

Chapter 2

The Gauss-Newton Filter

This chapter discusses the derivation of the Gauss-Newton filter (GNF) which is the focus of this thesis. The chapter also describes how the GNF is related to the minimum variance, the Cramer-Rao lower bond and the maximum likelihood. Two new types of estimators: the linear GN filter with process noise and the linear biased GN filter are then introduced. Finally some key issues such as the filter implementation, the effect of memory length on the filter performance as well as the filter convergence are briefly analyzed.

2.1 State space model based on non-linear differential equations

Consider the following autonomous, non-linear differential equation (DE) governing the process state:

$$DX(t) = F(X(t)) \tag{2.1}$$

in which F is a non-linear vector function of the state vector X describing a process, such as the position of a target in space and D the differential operator. The state vector presented here is deterministic as opposed to general case where the state is perturbed by random noise, commonly referred to as process noise. We will later show that this assumption does not restrict the filter from estimating non deterministic signals. We assume the observation scheme of the process is a non-linear function of the process state with expression:

$$Y(t) = G(X(t)) + v(t) \tag{2.2}$$

where G is a non-linear function of X and $v(t)$ is a random Gaussian vector. The goal is to estimate the process state from the given non-linear state models. For linear differential equations (DEs), the state transition matrix could be easily obtained. This, however, is not the case with non-linear DEs. Nevertheless, there is a procedure, based on local linearisation, that enables us to get around this obstacle, which we will now present.

2.1.1 The method of local linearisation

The solution of the DE gives rise to infinitely many trajectories that are dependent on the initial condition. However there will be one trajectory whose state

vector the filter will attempt to identify from the observations. We assume that there is a known nominal trajectory with state vector $\bar{X}(t)$ that has the following properties:

- $\bar{X}(t)$ satisfies the same DE as $X(t)$.
- $\bar{X}(t)$ is close to $X(t)$.

The above-mentioned properties result in the following expression:

$$X(t) = \bar{X}(t) + \delta X(t) \tag{2.3}$$

where $\delta X(t)$ is a vector of time-dependent functions that are small in relation to the corresponding elements of either $\bar{X}(t)$ or $X(t)$. The vector $\delta X(t)$ is called the *perturbation vector* and is governed by the following DE (the derivation is shown in Appendix A):

$$D(\delta X(t)) = A(\bar{X}(t))\delta X(t) \tag{2.4}$$

where $A(\bar{X}(t))$ is called a sensitivity matrix defined as follows:

$$A(\bar{X}(t)) = \left.\frac{\partial F(X(t))}{\partial (X(t))}\right|_{\bar{X}(t)} \tag{2.5}$$

.

Equation 2.4 is thus a linear DE, with a time varying coefficient and has the following transition equation:

$$\delta X(t+\zeta) = \Phi(t_n+\zeta, t_n, \bar{X})\delta X(t) \tag{2.6}$$

in which $\Phi(t_n+\zeta, t_n, \bar{X})$ is the transition matrix from time t_n to $t_n+\zeta$ (increment ζ). The transition matrix is governed by the following DE:

$$\frac{\partial}{\partial\zeta}\Phi(t_{n+\zeta}, t_n, \bar{X}) = A(\bar{X}(t_n+\zeta))\Phi(t_{n+\zeta}, t_n, \bar{X}) \tag{2.7}$$

with

$$\Phi(t_n, t_n, \bar{X}) = I \tag{2.8}$$

.

The transition matrix is a function of $\bar{X}(t)$ and can be evaluated by numerical integration and in order to fill the values of $A(\bar{X}(t_n+\zeta))$, $\bar{X}(t)$ has to be integrated numerically. In Chapter 3 we present a recursive algorithm that avoids the computation of the transition matrix. We have shown in this section that we can estimate the true state of process by estimating the perturbation vector, which is governed by a linear differential equation. The next task is to obtain a linear perturbation observation from the non-linear observation scheme.

2.1.2 The observation perturbation vector

In this section we will adopt the notation X_n and Y_n for $X(t_n)$ and $Y(t_n)$ respectively. We define a simulated noise free observation vector $\bar{Y}_n$ as follows:

$$\bar{Y}_n = G(\bar{X}_n) \tag{2.9}$$

.

Subtracting $\bar{Y}_n$ from the actual observation Y_n gives the *observation perturbation vector*:

$$\delta Y_n = Y_n - \bar{Y}_n \tag{2.10}$$

.

In appendix A.2 we show that the observation perturbation vector is related to the state perturbation vector as follows:

$$\delta Y_n = M(\bar{X}_n)\delta X_n + v_n \tag{2.11}$$

where $M(\bar{X}_n)$ is the Jacobean matrix of G, evaluated at $\bar{X}_n$. The matrix is also called the *observation sensitivity matrix* and is defined as follows:

$$M(\bar{X}_n) = \left.\frac{\partial F(X_n)}{\partial (X_n)}\right|_{\bar{X}_n} \tag{2.12}$$

.

We now examine the sequence of observations.

2.1.3 Sequence of observation

We assume that $L+1$ observation are obtained with time stamps $t_n, t_{n-1}, ..., t_{n-L}$. Theses observations are assembled as follows:

$$\begin{bmatrix} \delta Y_n \\ \delta Y_{n-1} \\ . \\ . \\ . \\ \delta Y_{n-L} \end{bmatrix} = \begin{bmatrix} M(\bar{X}_n)\delta X_n \\ M(\bar{X}_{n-1})\delta X_{n-1} \\ . \\ . \\ . \\ M(\bar{X}_{n-L})\delta X_{n-L} \end{bmatrix} + \begin{bmatrix} v_n \\ v_{n-1} \\ . \\ . \\ . \\ v_{n-L} \end{bmatrix} \tag{2.13}$$

Using the relationship:

$$\delta X_m = \Phi(t_m, t_n, \bar{X})\delta X_n \tag{2.14}$$

then, substituting Equation 2.15 the observation sensitivity equation can be written as:

$$\delta \mathbf{Y}_n = \mathbf{T}_n \delta X_n + \mathbf{V}_n \tag{2.15}$$

in which $\mathbf{T}_n$, the *total observation matrix* is defined as follows:

$$\mathbf{T}_n = \begin{bmatrix} M(\bar{X}_n) \\ M(\bar{X}_{n-1})\Phi(t_{n-1}, t_n; \bar{X}) \\ . \\ . \\ . \\ M(\bar{X}_{n-L})\Phi(t_{n-L}, t_n; \bar{X}) \end{bmatrix} \tag{2.16}$$

.

Where $M(\bar{X}_n)$ is the Jacobian of G computed at nominal trajectory $\bar{X}_n$ and $\Phi(t_{n-i}, t_n)$ the transition matrix representing the backward transition from time t_n to t_{n-i}.

2.1.4 The Hessian matrix

We can construct the error function(also called the cost function) as follows:

$$E_n(\delta X_n) = (\delta \mathbf{Y}_n - \mathbf{T}_n \delta X_n)^T \mathbf{R}_n^{-1} (\delta \mathbf{Y}_n - \mathbf{T}_n \delta X_n) \tag{2.17}$$

where δX_n is the perturbation vector or the step in the Gauss-Newton iterative method, and $\mathbf{R}_n^{-1}$ is a block diagonal weight matrix, also called the *least squares weight matrix*. However if we define R_n as the covariance matrix of the error vector v_n, then $\mathbf{R}_n^{-1}$ is expressed as:

$$\mathbf{R}_n^{-1} = \begin{bmatrix} R_n^{-1} & 0 & . & . & . & 0 \\ 0 & R_{n-1}^{-1} & & & & . \\ . & & . & & & . \\ . & & & . & & . \\ . & & & & . & \\ 0 & . & . & . & 0 & R_{n-L}^{-1} \end{bmatrix} \tag{2.18}$$

We would want to find δX_n that minimizes the cost function. After expansion of the cost function it is evident that the gradient and the Hessian are as follows:

$$E^{'}(\delta X_n) = -2\mathbf{T}_n^T\mathbf{R}^{-1}\delta\mathbf{Y}_n + 2\mathbf{T}_n^T\mathbf{R}_n^{-1}\mathbf{T}_n\delta X_n \tag{2.19}$$

$$E^{''}(\delta X_n) = 2\mathbf{T}_n^T\mathbf{R}_n^{-1}\mathbf{T}_n \tag{2.20}$$

The Hessian is independent of δX_n and is symmetric and if $\mathbf{T}_n$ is full rank then it is also positive definite. Hence $E(\delta X_n)$ has unique minimiser defined as follows:

$$\delta\hat{X}_n = (\mathbf{T}_n^T\mathbf{R}_n^{-1}\mathbf{T}_n)^{-1}\mathbf{T}_n^T\mathbf{R}_n^{-1}\delta\mathbf{Y}_n \tag{2.21}$$

The covariance matrix associated to the estimation is

$$\hat{S}_n = (\mathbf{T}_n^T\mathbf{R}_n^{-1}\mathbf{T}_n)^{-1} \tag{2.22}$$

2.2 Consistency of The Gauss-Newton filter

In this section we use the Linear approach to discuss the consistency of the Gauss-Newton filter with respect to Cramer-Rao and the maximum likelihood. The Linear form of the Gauss-Newton filter, referred to as Gauss-Aitken filter in [3, 4] is as follows:

$$\hat{X}_n = (\mathbf{T}_n^T \mathbf{R}_n^{-1} \mathbf{T}_n)^{-1} \mathbf{T}_n^T \mathbf{R}_n^{-1} \mathbf{Y}_n \tag{2.23}$$

With covariance matrix :

$$\hat{S}_n = (\mathbf{T}_n^T \mathbf{R}_n^{-1} \mathbf{T}_n)^{-1} \tag{2.24}$$

2.2.1 Cramer-Rao

Recalling the observation vector equation for linear equation:

$$\mathbf{Y}_n = \mathbf{T}_n X_n + \mathbf{V}_n \tag{2.25}$$

The likelihood function of X_n given $\mathbf{Y}_n$ is:

$$\ell_n = (2\pi)^{-k/2} |\mathbf{R_n}|^{-1/2} \exp -\frac{1}{2} \left((\mathbf{Y}_n - \mathbf{T}_n X_n)^T \mathbf{R}_n^{-1} (\mathbf{Y}_n - \mathbf{T}_n X_n) \right) \tag{2.26}$$

Taking the logarithm of the likelihood function gives the following expression:

$$\ln(\ell_n) = -\frac{1}{2} (k \ln (2\pi) + \ln |\mathbf{R}_n|) - \frac{1}{2} \left((\mathbf{Y}_n - \mathbf{T}_n X_n)^T \mathbf{R}_n^{-1} (\mathbf{Y}_n - \mathbf{T}_n X_n) \right) \tag{2.27}$$

Differentiating the log-likelihood function by X_n results in:

$$\frac{\partial^2 \ln(\ell_n)}{\partial X_n^T \partial X_n} = -\mathbf{T}_n^T \mathbf{R}_n^{-1} \mathbf{T}_n \tag{2.28}$$

The Cramer-Rao lower bound is defined as the mean of the second derivation of the logarithm of the likelihood function. It is therefore defined by the following expression:

$$S_{CR} = (-\varepsilon \frac{\partial^2 \ln(\ell_n)}{\partial X_n^T \partial X_n}])^{-1} \tag{2.29}$$

becoming:

$$S_{CR} = (\mathbf{T}_n^T \mathbf{R}_n^{-1} \mathbf{T}_n)^{-1} \tag{2.30}$$

Which is the Covariance matrix of the Gauss-Newton filter. Therefore we can conclude the Gauss-Newton filter is Cramer-Rao consistent. In fact in [3] Morrison demonstrates that the minimum variance algorithm has a Cramer-Rao Lower bound for any distribution of the multivariate V_n.

2.2.2 The maximum likelihood

The Gauss-Newton filter is derived from the Minimum variance algorithm. In this section we show that the filter obeys the maximum likelihood theory. Referring to the likelihood function ℓ_n defined in Equation 2.27, the method of maximum likelihood is to find X_n such that ℓ_n is maximized. This in turn means to find

X_n that minimizes exponent, namely:

$$(\mathbf{Y}_n - \mathbf{T}_n X_n)^T \mathbf{R}_n^{-1} (\mathbf{Y}_n - \mathbf{T}_n X_n) \tag{2.31}$$

Which is the cost function defined in Equation 2.20. Therefore when the errors in the multivariate are Gaussian the Gauss-Newton filter is consistent with the maximum likelihood estimate.

2.3 Process Noise

In this section we demonstrate how to incorporate the process noise statistics into the GN filter. For simplicity we consider a linear state space model:

$$X_{n+1} = \Phi(t_{n+1}, t_n) X_n + w_n \tag{2.32}$$

$$Y_n = M X_n + v_n \tag{2.33}$$

Where W_n is the process noise assumed to be Gaussian with covariance matrix Q_n and v_n the observation noise with covariance matrix R_n. Using 2.32, the state

equation can be expanded over the memory length as follow:

$$\begin{aligned}
X_n &= X_n \\
X_{n-1} &= \Phi(t_{n-1}, t_n)X_n + w_{n-1} \\
X_{n-2} &= \Phi(t_{n-2}, t_n)X_n + w_{n-2} \\
&\vdots \\
X_{n-L} &= \Phi(t_{n-L}, t_n)X_n + w_{n-L}
\end{aligned} \tag{2.34}$$

The corresponding measurement equation is therefore:

$$\begin{aligned}
Y_n &= MX_n + v_n \\
Y_{n-1} &= \Phi(t_{n-1}, t_n)MX_n + Mw_{n-1} + v_{n-1} \\
Y_{n-2} &= \Phi(t_{n-2}, t_n)MX_n + Mw_{n-2} + v_{n-2} \\
&\vdots \\
Y_{n-L} &= \Phi(t_{n-L}, t_n)MX_n + Mw_{n-L} + v_{n-L}
\end{aligned} \tag{2.35}$$

which can be written as a function of the $\mathbf{T}$ as follows:

$$\mathbf{Y}_n = \mathbf{T}_n X_n + \mathbf{V}_n \tag{2.36}$$

We therefore have a basis for minimum variance estimation with $\mathbf{V}_n$ the new long vector of random vectors defined as:

$$\mathbf{V}_n = \begin{bmatrix} v_n \\ w_{n-1} + v_{n-1} \\ w_{n-2} + v_{n-2} \\ . \\ . \\ . \\ w_{n-L} + v_{n-L} \end{bmatrix} \tag{2.37}$$

with covariance matrix:

$$\mathbf{R}_n = \begin{bmatrix} R_n & 0 & . & . & . & 0 \\ 0 & R_{n-1} + MQ_{n-1}M^T & & & & . \\ . & & . & & & . \\ . & & & . & & . \\ . & & & & . & \\ 0 & . & . & . & 0 & R_{n-L+1} + MQ_{n-L+1}M^T \end{bmatrix} \tag{2.38}$$

From 2.36 it is evident that the estimator equation remains unchanged. However the weight matrix is now a result of the combined measurement and process noise covariance matrices. For simplicity we assume unchanged statistics of the observation noise and the process noise, that is, $R_n = R$ and $Q_n = Q$.

2.3.1 Error covariance consistency and the role of Q matrix

One of the most talked about issues with the Kalman filter is the process covariance matrix which is introduced to limit the sensitivity of the filter to processes with driving noise. The process noise is part of the dynamic and therefore it is not observable. This makes the presence of the Q matrix in the filter equation arbitrary. The Q matrix can create an error covariance inconsistency, that is, the actual error in the estimation does not correspond to what the covariance matrix shows. Here we mathematically investigate the effect of the Q matrix.

The minimum cost function is defined in term of the Q matrix as follows:

$$E_n(Q) = \left(\mathbf{Y}_n - \mathbf{T}_n\hat{X}_n\right)^T \mathbf{R}_n\left(Q\right)^{-1}\left(\mathbf{Y}_n - \mathbf{T}_n\hat{X}_n\right) \tag{2.39}$$

Equation 2.39 can be expanded to arrive at the following expressions:

$$\begin{aligned} E_n(Q) &= \mathbf{Y}_n^T\mathbf{R}_n\left(Q\right)^{-1}\mathbf{Y} - \hat{X}_n^T\left(\mathbf{T}_n^T\mathbf{R}_n\left(Q\right)^{-1}\mathbf{T}_n\right)\hat{X}_n \\ &= trace\left(\mathbf{R}_n\left(Q\right)^{-1}\mathbf{Y}_n^T\mathbf{Y}_n\right) - trace\left(\left(\mathbf{T}_n^T\mathbf{R}_n\left(Q\right)^{-1}\mathbf{T}_n\right)\hat{X}_n^T\hat{X}_n\right) \end{aligned} \tag{2.40}$$

The mean of the minimum cost function is obtained as follows:

$$\begin{aligned} \varepsilon(E_n(Q)) &= trace\left(\mathbf{R}_n\left(Q\right)^{-1}\varepsilon\left(\mathbf{Y}_n^T\mathbf{Y}_n\right)\right) - trace\left(\left(\mathbf{T}_n^T\mathbf{R}_n\left(Q\right)^{-1}\mathbf{T}_n\right)\varepsilon\left(\hat{X}_n^T\hat{X}_n\right)\right) \\ &= trace\left(\mathbf{R}_n\left(Q\right)^{-1}\mathbf{R}_n\right) - trace\left(\left(\mathbf{T}_n^T\mathbf{R}_n\left(Q\right)^{-1}\mathbf{T}_n\right)\left(\mathbf{T}_n^T\mathbf{R}_n\left(Q\right)^{-1}\mathbf{T}_n\right)^{-1}\right) \\ &= trace\left(I + (L-1)(R + MQM^T)^{-1}R\right) - N_X \end{aligned} \tag{2.41}$$

where N_X is length of the state vector. The final expression of the minimum cost function is therefore as follows:

$$\varepsilon(E_n(Q)) = (L-1)trace\left((R+MQM^T)^{-1}R\right) + N_Y - N_X \tag{2.42}$$

where N_Y is the length of the single observation vector Y. Given that Q is positive definite the following inequality:

$$trace\left((R+MQM^T)^{-1}R\right) < trace\left((R)^{-1}R\right)$$

is always true A.3. This leads to $\varepsilon(E_n(Q)) < LN_L - N_X$ or $\frac{\varepsilon(E_n(Q))}{LN_L - N_X} < 1$. This means that the normalized cost function $\frac{\varepsilon(E_n(Q))}{LN_L - N_X}$ is reduced by the introduction of the Q matrix. The Q matrix here acts as a bias matrix that reduces the mean squared error.

Since $(R+MQM^T)^{-1} < R^{-1}$ from Equation 2.43, the following matrix inequality holds:

$$\mathbf{R}\,(Q)_n^{-1} < \mathbf{R}_n^{-1} \tag{2.43}$$

which leads to:

$$\left(\mathbf{T}_n^T\mathbf{R}\,(Q)_n^{-1}\,\mathbf{T}_n\right) < \left(\mathbf{T}_n^T\mathbf{R}_n^{-1}\mathbf{T}_n\right) \tag{2.44}$$

Therefore the Q matrix reduces the amount of information in the Fisher information matrix. In other words, the variance or the standard error is increased by the presence of the Q matrix. We have demonstrated here that the Q matrix reduces the mean squared error but increases the variance of the error. This result explains mathematically the error covariance inconsistency of the Kalman filter

observed by Morrison in [3]. Unlike the Kalman filter this estimator covariance will not be reused in future estimation therefore it can be adjusted (or shrunk) to reflect the reduction in the mean squared error.

2.4 A Biased Estimator Version

Bias estimators have gained considerable interest in radar literature [11]. The Bias is introduced as a mean to minimize the mean squared error (MSE). This approach lowers the Cramer-Rao lower bound which is very fascinating[11]. Here we apply a similar approach to obtain a version of the linear Gauss-Newton filter with bias.

2.4.1 The Exactness constraint

If we defined the filter matrix as:

$$\mathbf{W}_n = (\mathbf{T}_n^T \mathbf{R}_n^{-1} \mathbf{T}_n)^{-1} \mathbf{T}_n^T \mathbf{R}_n^{-1} \tag{2.45}$$

then from 2.23 we have:

$$\hat{X}_n = \mathbf{W}_n \mathbf{Y}_n \tag{2.46}$$

The exactness constraint is defined as $\mathbf{W}_n \mathbf{T}_n = I$, where I is an identity matrix.

2.4.2 The Mean Squared Error(MSE)

As by Kai et al. in [11], the biased estimate $\hat{X}_b$ and the minimum variance estimate $\hat{X}_n$ are related as follows:

$$\hat{X}_b = (I + B)\hat{X}_n \tag{2.47}$$

where B is the Bias matrix to be determined. The error between the biased estimate and the true state X_n is:

$$\hat{X}_b - X_n = (I + B)\hat{X}_n - X_n = (I + B)\mathbf{W}_n\mathbf{Y}_n - X_n \tag{2.48}$$

After further development, considering 2.15 and the exactness constraint, the following is obtained:

$$\hat{X}_b - X_n = BX_n + (I + B)\mathbf{W}_n\mathbf{V}_n \tag{2.49}$$

Therefore the MSE is:

$$MSE(B) = E(\left\|\hat{X}_b - X_n\right\|^2) = X_n^T B^T B X_n + trace\left(\left[(I + B)\,\mathbf{W}_n\right]^T \mathbf{R}\left[(I + B)\,\mathbf{W}_n\right]\right) \tag{2.50}$$

Equation 2.50 shows the MSE has two components, the Bias $X_n^T B^T B X_n$ and the Variance $trace\left(\left[(I + B)\,\mathbf{W}_n\right]^T \mathbf{R}\left[(I + B)\,\mathbf{W}_n\right]\right)$. It is evident that the MSE is minimized for negative values of B. The Bias increases as B departs from zeros while the the variance decreases. The ideal solution is to find B that minimizes the overall MSE.

2.4.3 Obtaining the bias matrix

The value of B that minimizes the MSE is obtained through differentiation in terms of B of the MSE expression in Equation 2.50 and equating the answer to zero:

$$\begin{aligned} B &= -\mathbf{W}_n\mathbf{R}_n{\mathbf{W}_n}^T(X_n^T X_n + \mathbf{W}_n\mathbf{R}_n{\mathbf{W}_n}^T)^{-1} \\ &= -(\mathbf{T}_n^T\mathbf{R}_n^{-1}\mathbf{T}_n)^{-1}(X_n^T X_n + (\mathbf{T}_n^T\mathbf{R}_n^{-1}\mathbf{T}_n)^{-1})^{-1} \end{aligned} \tag{2.51}$$

The optimum bias vector is therefore a function of the target true state vector. We propose the use of the predicted state for an approximation of the B matrix. A detailed analysis of such an approach is beyond the scope of this work.

2.5 The Iterative implementation

The GNF is a non-linear optimization method and therefore is iterative. From starting point $\bar{X}_n$, the method produces series of $\hat{X}_n$ which hopefully converges to the local minimizer of a given cost function. The positive definiteness of the Hessian ensures convergence towards the local minimizer. The GN has a faster convergence rate in the final stage of the iteration, therefore one must ensure the initial guess is close enough to the local minimizer. Furthermore the $\mathbf{T}$ matrix is recomputed at each iteration. The iterative implementation of the GNF is summarized as follows:

1 obtain the nominal trajectory $\bar{X}_n$

2 estimate $\delta\hat{X}_n$ from Equation 2.22

3 make $\bar{X}_n = \hat{X}_n$ and repeat 2 until $\delta\hat{X}_n$ is insignificant enough

A detailed implementation algorithm is presented in the appendix page.

2.6 The Effect of the Filter memory length

The Filter memory length offers some advantages that we now describe.

2.6.1 Filter initialization

The GNF does not require an initial covariance matrix as is the case with the the EKF. Therefore, it is practically self-initializing . One aspect that improves the filter's robustness is its ability to adaptively change the memory length. For track initialization the GNF can start with a minimum memory length (typically 4) and then increase incrementally for better estimation. Such method enables a self-initializing capability of the filter. All that is required is to have the initial guess close enough to the local minimizer to ensure faster convergence.

2.6.2 Non-linearity

The other advantage the memory length offers is the ability to sustain its performance when abrupt changes occur. Such changes could be random acceleration in the state vector or some deterministic non-linearity such as manoeuvres. The

memory length can be changed adaptively for smooth transition between manoeuvres. The Master Control Algorithm (MCA) developed by Morrison uses such a strategy for tracking highly manoeuvring targets. The availability of the residuals for a fixed memory length allows online verification of the bias free nature of the filter as well as the error covariance consistency test.

2.7 Convergence issues of the Gauss-Newton filter

The GNF has in general a linear convergence rate which could be quadratic in some situations. The Convergence rate becomes super linear when the estimate gets closer to the local minimizer. Therefore it is important to have the initial guess close enough to the local minimizer. This is generally not problematic in radar target tracking where predicted target position are used as initial guess. Nonetheless, when incorrect predictions are made such as in the case of dynamic target change, the GNF may fail to converge. In Chapter 4, a method that combines the Gauss-Newton method and the steepest descent is adopted to resolve such difficulty.

2.7.1 Rank deficiency

We have shown in Section 2.1.4 that the necessary condition for having a unique solution to the cost function, which is also the local minimizer, is to have a positive definite Hessian. However, the Hessian can lose the positive definite property

Task	Final Size	Complexity
$\mathbf{R}^{-1}$	$mL \times mL$	$\mathcal{O}(Lm^2)$
$\mathbf{T}$	$mL \times k$	$\mathcal{O}(Ln^2 + Lmk^2)$
$\mathbf{TR}^{-1}\mathbf{T}$	$n \times k$	$\mathcal{O}(Lmk^2 + km^2L^2)$
$(\mathbf{TR}^{-1}\mathbf{T})^{-1}$	$k \times k$	$\mathcal{O}(k^3)$
$\mathbf{TR}^{-1}\mathbf{Y}$	$k \times 1$	$\mathcal{O}(km^2L^2 + Lmk)$
$\hat{\mathbf{X}}$	$k \times 1$	$\mathcal{O}(k^2)$

Table 2.1: Computational complexity

due to the nature of the observation scheme or due to numerical approximation errors. When these situation occurs the filter will fail to converge. This problem is not unique to the GNF, as it is shown in Chapter 3 the inverse of the Hessian is also the covariance matrix of IEKF.

2.7.2 Computational Complexity

The memory nature of GNF makes it computationally intensive. Table **??**shows the computational complexity involved in each stage of the filter operation. The major contributors are the computation of the $\mathbf{T}$ matrix and the Hessian matrix $(\mathbf{T}_n^T\mathbf{R}_n^{-1}\mathbf{T}_n)$. For a single cycle of the filter the required computation is of the order $\mathcal{O}(km^2L^2)$, where k is the size of the state vector and m the size of measurement vector. This result is obtained assuming $L \gg k$. Therefore the GN computation requirements increase quadratically with the memory length. The computation load of the GNF is very significant when compared to the Kalman filter or the recursive GNF which has a complexity of $\mathcal{O}(k^3)$ or lower (when matrix inversion is avoided). In the next chapter a recursive form of the GNF is proposed to overcome the computation burden.

Conclusion

This chapter provides a general overview of the Gauss-Newton filter. The derivation of the filter equations from first principles where described. Our contributions in this chapter are:

- The incorporation of the process noise in the filter equation which shed some light on the error covariance inconsistency seen in the Kalman filter.
- The derivation of bias tracking filter based on the linear Gauss-Newton filter which can offer superior mean squared error.

These contributions are not core to our research and are therefore included in this chapter to highlight the key attributes of the Gauss-Newton filter. The next two chapters present the main contribution of this thesis.

Chapter 3

The recursive Gauss-Newton Filter

In Chapter 2, we arrived at a form of a filter that uses the minimum variance estimation initiated by Gauss and the local linearisation technique championed by Newton to estimate the estate of the process from the non-linear observation scheme. This filter is called the Gauss-Newton filter (GNF) and is described in detail in Morrison's work [3, 4]. The GNF has been successfully implemented in some practical applications:

[10] showing strong stability. The memory nature of the filter has made it unattractive to researchers in the past, and even now, challenging [6] . However recent developments have presented the recursive form of the linear least-squares for state space model [13]. We derive a recursive form of GNF using a similar approach to M. B. Malik [13].

3.1 Recursive vs. nonrecursive filtering

Recursive and non recursive are two fundamental type of filtering. In the non-recursive filtering, immediately after scan n, the accumulated observations are submitted to the filter and all are processed simultaneously to produce the filter estimate. None of the previous filter outputs is used to produce that estimate.

In the recursive filtering situation, the observation received at scan n is combined with estimate produced at scan $n-1$ to produce estimate at scan n.

Both recursive and nonrecursive filters can be made iterative. The estimate at scan n is reused a number of times with the same observation (at or up to scan n) to produce a refined estimate at scan n. Algorithms 3 and 2 show the implementation of the iterative nonrecursive and the iterative recursive algorithms respectively.

We now move to derive the *Recursive Gauss Newton Filter* in the next section.

3.2 The Recursive Gauss-Newton filter

To obtain the recursive form, we use an approach similar to M. B. Malik in [13].

3.2.1 Preliminaries

Before we derive a recursive form of the GNF filter, we rewrite the expression of $\mathbf{T}_n$ using the backward differentiation:

$$\Phi(t_{n-L}, t_n, \bar{X}) = A(\bar{X}_{n-L})^{-1}\Phi(t_{n-L+1}, t_n, \bar{X}) \tag{3.1}$$

the expression is thus:

$$\mathbf{T}_n = \begin{bmatrix} M_0 \\ M_1A_1 \\ M_2A_2 \\ . \\ . \\ . \\ M_LA_L \end{bmatrix} \tag{3.2}$$

where

$$A_L = \prod_{i=1}^{L} A(\bar{X}_{n-i})^{-1} \tag{3.3}$$

and

$$M_L = M(\bar{X}_{n-L}) \tag{3.4}$$

with

$$A_0 = I \tag{3.5}$$

Suppose that the observations start arriving at $n = 0$ and that all initial values of the filter are available. In in order to maintain the filter adaptiveness, a weight matrix function using a fading parameter $\lambda < 1$ is adopted, and is defined as

follows:

$$\mathbf{R}_n^{-1} = \begin{bmatrix} R^{-1} & 0 & . & . & . & 0 \\ 0 & \lambda R^{-1} & & & & . \\ . & & . & & & . \\ . & & & . & & . \\ . & & & & . & \\ 0 & . & . & . & 0 & \lambda^n R^{-1} \end{bmatrix} \tag{3.6}$$

. where R^{-1} is the inverse of the observation covariance matrix R. The following, further definitions are adopted:

$$W_n = \mathbf{T}_n^T \mathbf{R}_n^{-1} \mathbf{T}_n \tag{3.7}$$

$$\xi_n = \mathbf{T}_n^T \mathbf{R}_n^{-1} \delta \mathbf{Y} \tag{3.8}$$

Resulting in:

$$\delta \hat{X}_n = W_n^{-1} \xi_n \tag{3.9}$$

In the next section, the recursive update of the perturbation vector is demonstrated.

3.2.2 The recursive update of W_n

Using Equation (3.2) and the definitions in equations (4.7) and (3.6) we have:

$$W_n = \sum_{j=1}^{L} \lambda^j R^{-1} \prod_{i=1}^{j} A(\bar{X}_{n-i})^{-T} M(\bar{X}_{n-j})^T \times M(\bar{X}_{n-j}) \prod_{i=0}^{j} A(\bar{X}_{n-i})^{-1} + M(\bar{X}_n)^T R^{-1} M(\bar{X}_n) \tag{3.10}$$

and

$$W_{n-1} = \sum_{j=1}^{L-1} \lambda^j R^{-1} \prod_{i=1}^{j} A(\bar{X}_{n-1-i})^{-T} M(\bar{X}_{n-1-j})^T \times^{-1} M(\bar{X}_{n-1-j}) \prod_{i=0}^{j} A(\bar{X}_{n-1-i})^{-1} + M(\bar{X}_{n-1})^T R^{-1} M(\bar{X}_{n-1}) \tag{3.11}$$

Kalaman filter Comparing equations (3.10) and (3.11) the following recursive equation is obtained:

$$W_n = \lambda A(\bar{X}_{n-1})^{-T} W_{n-1} A(\bar{X}_{n-1})^{-1} + M(\bar{X}_n)^T R^{-1} M(\bar{X}_n) \tag{3.12}$$

which is the discrete, quadratic, Lyapunov, difference equation.

3.2.3 The recursive form of ξ_n

Using equations (3.2) (3.6) (4.8) ξ_n can be expressed as:

$$\xi_n = \sum_{j=0}^{L} \lambda^j R^{-1} \prod_{i=1}^{j} A(\bar{X}_{n-i})^{-T} M(\bar{X}_{n-j})^T \delta Y_{n-j} + M(\bar{X}_n)^T R^{-1} \delta Y_n \tag{3.13}$$

and

$$\xi_{n-1} = \sum_{j=1}^{L-1} \lambda^j R^{-1} \prod_{i=1}^{j} A(\bar{X}_{n-1-i})^{-T} M(\bar{X}_{n-1-j})^T \times \delta Y_{n-1-j} + M(\bar{X}_{n-1})^T R^{-1} \delta Y_{n-1} \tag{3.14}$$

Comparing equations (3.13) and (3.14) the following recursive equation is obtained:

$$\xi_n = \lambda A(\bar{X}_{n-1})^{-T} \xi_{n-1} + M(\bar{X}_n)^T R^{-1} \delta Y_n \tag{3.15}$$

3.2.4 The Basic form of the recursive Gauss-Newton filter

The recursive Gauss-Newton can be summarized in three basic equations:

$$W_n = \lambda A(\bar{X}_{n-1})^{-T} W_{n-1} A(\bar{X}_{n-1})^{-1} + M(\bar{X}_n)^T R^{-1} M(\bar{X}_n) \tag{3.16}$$

$$\xi_n = \lambda A(\bar{X}_{n-1})^{-T} \xi_{n-1} + M(\bar{X}_n)^T R^{-1} \delta Y \tag{3.17}$$

$$\delta \hat{X}_n = W_n^{-1} \xi_n \tag{3.18}$$

As discussed in Chapter 2, W_n is the Fisher information matrix. Here new information is added during recursion while older information is forgotten, just as in the non recursive filter. These basic recursive equations can be organized in different ways.

3.2.5 The recursive update of W_n^{-1}

It can be shown through matrix inversion lemma [14, 13] that $\mathbf{W}_n^{-1}$ can be obtained from 4.9 as follows:

$$\begin{aligned} W_n^{-1} &= \lambda^{-1}A(\bar{X}_n)W_{n-1}^{-1}A(\bar{X}_{n-1})^T \\ &\quad -\lambda^{-2}A(\bar{X}_{n-1})W_{n-1}^{-1}A(\bar{X}_{n-1})^T M(\bar{X}_n)^T \\ &\quad \times[R+\lambda^{-1}M(\bar{X}_n)A(\bar{X}_{n-1})W_{n-1}^{-1}A(\bar{X}_{n-1})^T M(\bar{X}_n)^T]^{-1} \\ &\quad \times M(\bar{X}_n)A(\bar{X}_{n-1})W_{n-1}^{-1}A(\bar{X}_{n-1})^T \end{aligned} \tag{3.19}$$

If we define:

$$\begin{aligned} K_n &= \lambda^{-1}A(\bar{X}_{n-1})W_{n-1}^{-1}A(\bar{X}_{n-1})^T M(\bar{X}_n)^T \\ &\quad \times[R+\lambda^{-1}M(\bar{X}_n)A(\bar{X}_{n-1})W_{n-1}^{-1}A(\bar{X}_{n-1})^T M(\bar{X}_n)^T]^{-1} \end{aligned} \tag{3.20}$$

then Equation 3.19 becomes:

$$W_n^{-1} = \lambda^{-1}[I-K_nM(\bar{X}_n)]A(\bar{X}_n)W_{n-1}^{-1}A(\bar{X}_{n-1})^T \tag{3.21}$$

which is exactly the same as the extended Kalman covariance matrix equation if we let $\lambda = 1$. K_n is the observer gain or the celebrated Kalman gain.

The perturbation vector equation

If we rearrange Equation 3.21 and we consider the expression in Equation 4.9 then the observer gain can be written as:

$$K_n \quad = \quad W_n^{-1} M(\bar{X}_n)^T R^{-1} \tag{3.22}$$

Considering the expression of ξ_n in 4.10, 3.9 becomes:

$$\delta\hat{X}_n = \lambda W_n^{-1} A(\bar{X}_{n-1})^{-T} \xi_{n-1} + K_n \delta Y_n \tag{3.23}$$

The Equation 3.23 highlights the difference between the EKF innovation vector (which is $K_n \delta Y_n$) and the RGNF which accounts for previous estimate in the innovation vector. The component $\lambda \mathbf{W}_n^{-1} A(\bar{X}_{n-1})^{-T} \xi_{n-1}$ is the difference between the two innovation vectors. We call this component the *differential perturbation vector (DPV)* and the next section will demonstrate its complete mathematical expression.

3.2.6 The differential perturbation vector(DPV)

From above DPV is defined as:

$$D_{\delta X_n} = \lambda W_n^{-1} A(\bar{X}_{n-1})^{-T} \xi_{n-1} \tag{3.24}$$

After replacing ξ_{n-1} by its expression obtained from 3.9, Equation 3.24 becomes:

$$D_{\delta X_n} = \lambda W_n^{-1} A(\bar{X}_{n-1})^{-T} W_{n-1} A(\bar{X}_{n-1})^{-1} A(\bar{X}_{n-1}) \delta\hat{X}_{n-1} \tag{3.25}$$

which, after considering Equation 4.9, becomes:

$$D_{\delta X_n} = W_n^{-1}[W_n^{-1} - M(\bar{X}_n)^T R^{-1} M(\bar{X}_n)] A(\bar{X}_{n-1}) \delta\hat{X}_{n-1} \tag{3.26}$$

We expand Equation 3.26 further and we use the expression of the gain in Equation 3.22 to obtain:

$$D_{\delta X_n} = [I - K_n M(\bar{X}_n)] \delta\hat{X}_{n/n-1} \tag{3.27}$$

If we define :

$$\delta X_{n/n-1} = \hat{X}_{n/n-1} - \bar{X}_n \tag{3.28}$$

Then the expression in 3.23 corresponds to the iterated extended Kalman filter (IEKF) when $\lambda = 1$.

3.2.7 Stability of the RGNF

The matrix W_n is the inverse of the covariance matrix of the filter and is therefore positive definite. The component $M(\bar{X}_n)^T R^{-1} M(\bar{X}_n)$ is positive semi definite. As a consequence the derived discrete Lyapunov equation in (4.9) is stable when $\rho(A(\bar{X}_n)^{-1}) < \lambda^{-1/2}$, where ρ stands for eigenvalues. Therefore the stability bound is expanded from 1 to $\lambda^{-1/2}$ [15, 16, 17]. For polynomial trajectories the eigenvalues of the sensitivity matrix are all equal to 1 but since the matrix is Jordan block diagonal the stability condition becomes $\rho(A(\bar{X}_n)^{-1}) \leq \lambda^{-1/2}$. This

means the EKF which share the same covariance matrix equation as the RGNF but has $\lambda = 1$ will not be stable if the condition on the sensitivity matrix is violated.

3.3 Adaptive forgetting factor control

In this section we describe how the forgetting factor can be varied adaptively to track manoeuvres. We follow Morrison's approach on using the sum of squared residual to adaptively vary the non-recursive GN memory length [3].

3.3.1 The sum of squared residuals

From the non recursive GN filter, the sum of the squared residuals (SSR) is defined as follows:

$$SSR = \left(\mathbf{Y}_n - \mathbf{T}_n\hat{\mathbf{X}}_n\right)^T \mathbf{R}_n^{-1} \left(\mathbf{Y}_n - \mathbf{T}_n\hat{\mathbf{X}}_n\right) \tag{3.29}$$

Which is the same as the minimized cost function. The random vector SSR has a pdf that is Chi-squared. The expectation of the SSR is

$$\varepsilon\{SSR\} = N_{\mathbf{Y}} - N_{\hat{X}} \tag{3.30}$$

where $N_{\mathbf{Y}}$ is the length of the total observation vector $\mathbf{Y}_n$ and $N_{\hat{X}}$ the length of the filter estimate $\hat{X}$. The Chi-squared pdf has $k = N_{\mathbf{Y}} - N_{\hat{X}}$ degree of freedom.

Since k is the mean of the SSR, the normalized $SSR(NSSR)$ is as follows:

$$NSSR = SSR/k \tag{3.31}$$

where, $NSSR$ is always less than 1. The SSR and the $NSSR$ are key to the adaptive variation of the GNF memory. We will use a similar approach for the recursive filter.

3.3.2 The filter memory Length v.s the forgetting factor

The minimum cost function or the sum of squared residual can be expressed as follows:

$$\begin{aligned} E_n(\lambda) &= \delta\mathbf{Y}_n^T\mathbf{R}_n(\lambda)^{-1}\delta\mathbf{Y}_n - \delta\hat{X}_n^T\mathbf{W}_n\delta\hat{X}_n \\ &= trace\left(\mathbf{R}_n(\lambda)^{-1}\delta\mathbf{Y}_n^T\delta\mathbf{Y}_n\right) - trace\left(\mathbf{W}_n\delta\hat{X}_n^T\delta\hat{X}_n\right) \end{aligned} \tag{3.32}$$

Where $\mathbf{R}_n(\lambda)^{-1}$ is the weight matrix defined in Equation 3.6. Note $\delta\mathbf{Y}_n$ has covariance matrix $\mathbf{R}_n$ which is defined in Chapter 2 by Equation 2.18. Taking the mean of the cost function in Equation 3.33:

$$\begin{aligned} \varepsilon(E_n(\lambda)) &= trace\left(\mathbf{R}_n(\lambda)^{-1}\varepsilon\left(\delta\mathbf{Y}_n^T\delta\mathbf{Y}_n\right)\right) - trace\left(\mathbf{W}_n\varepsilon\left(\delta\hat{X}_n^T\delta\hat{X}_n\right)\right) \\ &= trace\left(\mathbf{R}_n(\lambda)^{-1}\mathbf{R}_n\right) - trace\left(\mathbf{W}_n\mathbf{W}_n^{-1}\right) \\ &= trace\left(I + \lambda I + \lambda^2 I + ...\lambda^n I\right) - N_X \\ &\approx N_Y\frac{1}{1-\lambda} - N_X \end{aligned} \tag{3.33}$$

Therefore the filter memory is related to the forgetting factor by the following expression:

$$L \approx \frac{1}{1-\lambda} \tag{3.34}$$

3.3.3 The recursive sum of squared residuals

The SSR defined in 3.29 can be expanded as follows:

$$SSR = \tilde{Y}_n^T R_n^{-1} \tilde{Y}_n + \tilde{Y}_{n-1}^T R_{n-1}^{-1} \tilde{Y}_{n-1} + \tilde{Y}_{n-2}^T R_{n-2}^{-1} \tilde{Y}_{n-2} + ... + \tilde{Y}_{n-L}^T R_{n-L}^{-1} \tilde{Y}_{n-L} \tag{3.35}$$

where:

$$\tilde{Y}_n = \left(Y_n - M_n \hat{X}_n\right) \tag{3.36}$$

If the the cost function has a forgetting factor as it is the case with the derivation of the recursive filter then the expanded SSR would be :

$$SSR = \tilde{Y}_n^T R_n^{-1} \tilde{Y}_n + \lambda \tilde{Y}_{n-1}^T R_{n-1}^{-1} \tilde{Y}_{n-1} + \lambda^2 \tilde{Y} R_{n-2}^{-1} \tilde{Y}_{n-2} + ... + \lambda^{n-L} \tilde{Y}_{n-L}^T R_{n-L}^{-1} \tilde{Y}_{n-L} \tag{3.37}$$

Which can then be expressed recursively as:

$$RSSR_n = \lambda RSSR_{n-1} + \tilde{Y}_n^T R_n^{-1} \tilde{Y}_n \tag{3.38}$$

The $RSSR$ is an effective mean for checking bias which could be caused by target manoeuvre. It can also be used to adaptively vary the forgetting factor. The effective number of the square residuals used in computing the above sum is approximately $\frac{1}{1-\lambda}$. Therefore $RSSR_n$ has a chi-square distribution with degree

of freedom $\frac{1}{1-\lambda}N_Y - N_{\hat{X}}$. The $RSSR$ can be used directly for updating the forgetting factor if it is computed over a fixed length window, corresponding to a constant λ_w in 3.38. Alternatively, a normalized $RSSR$ ($NSSR$) can be computed if we wish to use the same λ. The $NSSR$ is as follows:

$$NSSR_n = RSSR_n/(\frac{1}{1-\lambda}N_Y - N_{\hat{X}}) \tag{3.39}$$

3.3.4 Updating the forgetting factor

The sum of squared residual can be used to update λ. When $RSSR_n$ is greater than an upper threshold value $RSSR_{max}$ a manoeuvre is declared and we should rapidly decrease λ for the filter to handle the disturbance. On the other hand when $RSSR_n$ is less than a lower threshold value $RSSR_{min}$ the fit is very good and we should slowly increase λ . For values of $RSSR_n$ between the two thresholds, the current value of λ is the most effective so it remains unchanged. Both the upper and and lower threshold can be set using the chi-square table. Figure 3.1 describes the updating algorithm

3.3.5 The Adaptive Switching and Control Algorithm

This section introduces the Adaptive Switching and Control Algorithm (ASCA), based on the forgetting factor, to switch the filter trajectory model for more accurate tracking. The ASCA discussed here is in the same frame of work as the Master Control Algorithm (MCA) proposed by Morrison for tracking manoeuvres with the non recursive Gauss-Newton filter. The MCA for the non recursive GN

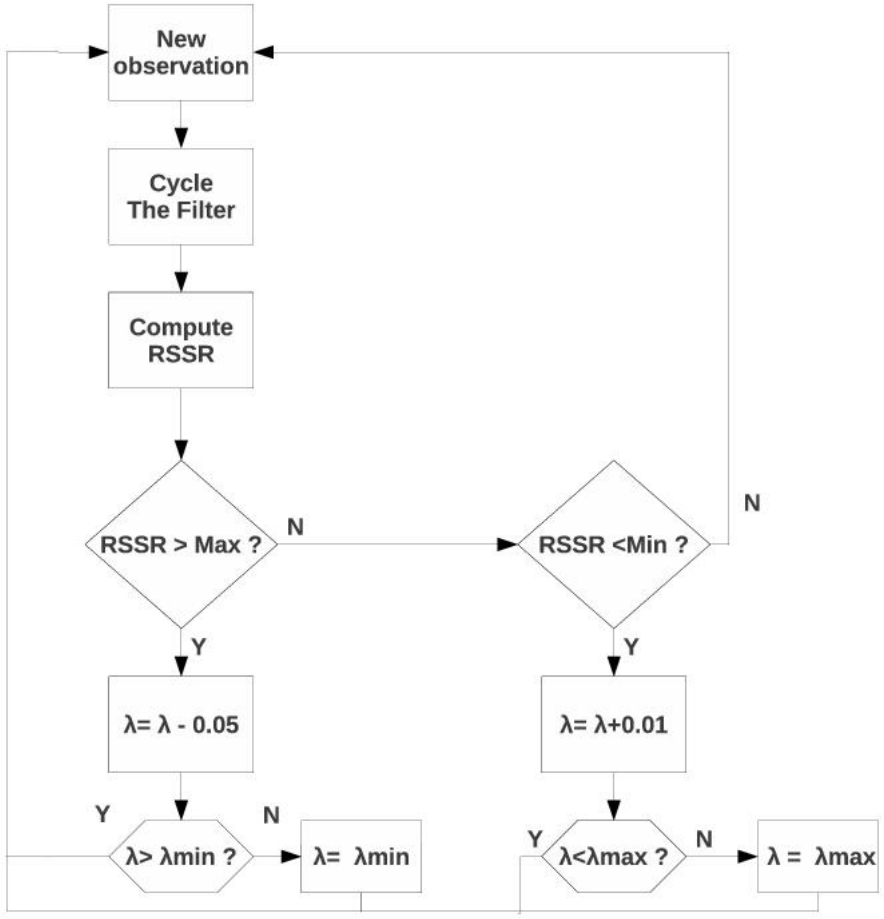

Figure 3.1: The adaptive forgetting factor algorithm

filter varies the filter memory adaptively with the aid of an adaptive sliding window. Referring to Figure 3.2 the operation of the forgetting factor based ASCA can be described as follows:

1. At start up a trajectory model is assumed for the filter. Usually a Degree 1 polynomial (constant velocity) model is chosen. The filter is then cycled using the selected trajectory model.

2. The RSSR is calculated to select an optimum value of λ, just as described in algorithm. If a manoeuvre is detected, then λ is decreased rapidly to a minimum value. If for that minimum value the RSSR is still above the threshold value then, we check for the number of times the filter cycled using the trajectory model. For the value of counter less than a threshold, we assume the filter is still initializing so the filter model is unchanged. On the other hand if the counter is more than the threshold value then the current model is not correct, therefore a new model is used. The counter is reset and the procedure is repeated. The counter threshold can be set by considering the memory length that is equivalent to the minimum forgetting factor ($L \approx \frac{1}{1-\lambda_{min}}$). Decreasing λ to a minimum before changing the filter model is essential for a smoother and faster settling of the filter after the change.

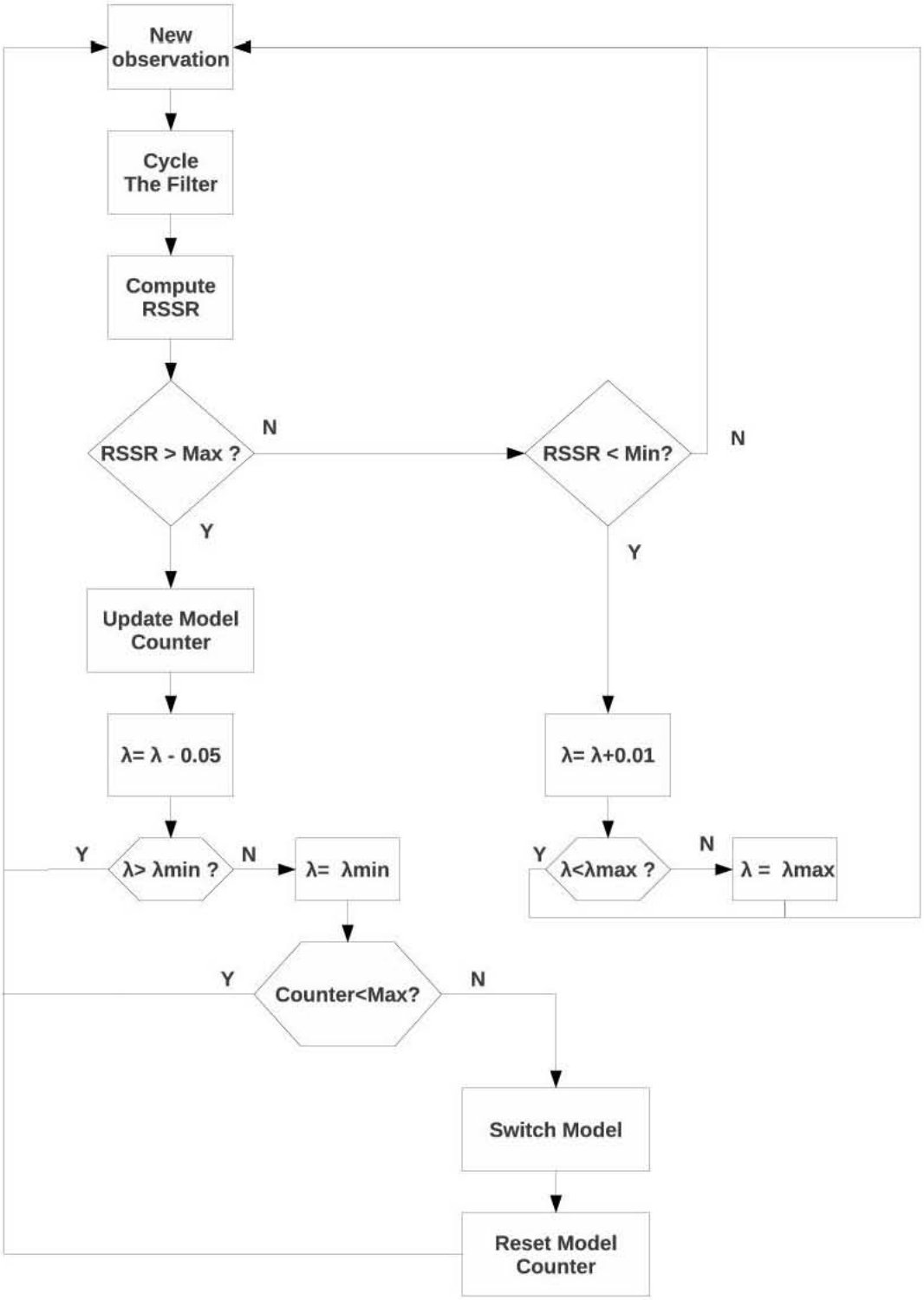

Figure 3.2: The Adaptive Switching and Control Algorithm (ASCA) based on the adaptive forgetting factor algorithm

3.4 Simulation

The objective of this simulation is to demonstrate the effectiveness of the ASCA algorithm. The derivation of the recursive Gauss-Newton filter presents us with multiple ways of implementing the filter. Any of the implementation algorithm can effectively be used in the ASCA. In this study we select the version of filter that is the iterated extended Kalman filter (IEKF) with forgetting factor. The choice of this algorithm is to demonstrate how ineffective the IEKF (when $\lambda = 1$) could be in tracking manoeuvres. Here we consider a vehicle that executes manoeuvres. During turn manoeuvres of known constant turn rate Ω, the vehicle dynamic model is:

$$X_n = \begin{bmatrix} 1 & \frac{\sin(\Omega T)}{\Omega} & 0 & -(\frac{1-\cos(\Omega T)}{\Omega}) \\ 0 & \cos(\Omega T) & 0 & -\sin(\Omega T) \\ 0 & \frac{1-\cos(\Omega T)}{\Omega} & 1 & \frac{\sin(\Omega T)}{\Omega} \\ 0 & \sin(\Omega T) & 0 & \cos(\Omega T) \end{bmatrix} X_{n-1} + v_n \tag{3.40}$$

where the state of the vehicle is $X_n = [x, \dot{x}, y, \dot{y}]$, with x, y the position coordinates and $\dot{x}$, $\dot{y}$ their corresponding velocity components. The process noise $v_n \sim \mathcal{N}(0, Q)$ with covariance matrix $Q = diag \begin{bmatrix} qBB^T & qBB^T \end{bmatrix}$ where,

$$BB^T = \begin{bmatrix} \frac{T^4}{4} & \frac{T^3}{2} \\ \frac{T^3}{2} & T^2 \end{bmatrix} \tag{3.41}$$

When the vehicle moves at a nearly constant velocity its dynamic model is:

$$X_n = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} X_{n-1} + v_n \tag{3.42}$$

The vehicle is observed by a radar located at the origin of the plane, capable of measuring the range r and the bearing angle θ. The measurement equation is therefore:

$$\begin{bmatrix} r_n \\ \theta_n \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2} \\ tan^{-1}(\frac{y}{x}) \end{bmatrix} + w_n \tag{3.43}$$

where the measurement noise is $w_k \sim \mathcal{N}(0, R)$ with covariance

$R = diag \begin{bmatrix} \sigma_r^2 & \sigma_\theta^2 \end{bmatrix}$. The following constants were used for data generation: $T = 1s$; $\Omega = -3^o s^{-1}$; $q = 0.1\text{m}^2\text{s}^{-4}$; σ_r=50m; $\sigma_\theta = 1$mrad.

The vehicle starts at true initial state X_n=[1000m, 50ms^{-1},-2000m, 20ms^{-1}] and moves at nearly constant velocity until $n = 300$, Then it executes a turn manoeuvre from time index $n = 301$ to $n = 600$. After the manoeuvre, the vehicle's velocity remains nearly constant from $n = 600$ to $n = 800$. The values $R = diag[50^2\ 10^{-6}]$; $T = 1$s; $W_{0/0}^{-1} = diag[50^2\ 10\ 50^2\ 10]$; $X_{0/0} = [1000\ 50\ -2000\ 20]^T$; $\lambda_{initial} = 0.9$; $\lambda_w = 0.9$; $\lambda_{max} = 0.99$; $\lambda_{min} = 0.3$; $count_{max} = 1/(1 - \lambda_w)$; $RSSR_{max} = 40$; $RSSR_{min} = 20$ were maintained throughout the simulation. The experiment was repeated for 50 Monte Carlo runs and the root mean squared error (RMSE) is used as the performance metric. The position

RMSE is computed using the following expression:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}((x_n^i - \hat{x_n})^2 + (y_n^i - \hat{y_n})^2)} \tag{3.44}$$

where (x_n^i, y_n^i) and $(\hat{x_n}, \hat{y_n})$ are true and estimated position coordinates respectively. The velocity RMSE is computed similarly. In Figures 3.3 and 3.4 we present The RMSE for different values of λ when the filter uses a single nearly constant speed model through out the simulation. It is evident from these results that for $\lambda = 1$ or closer the filter diverges when the filter model and the actual trajectory do not match up. While lower values of λ allows the filter to handle manoeuvres. The results also show that having higher values of λ improves the RMSE but reduces the filter adaptability. Figure 3.5 and 3.6 demonstrate the effectiveness of the ASCA. Both the position and the velocity RMSE are kept low by the switching mechanism. Figure 3.8 and 3.7 highlight the correlation between the RSSR and λ. When a manoeuvre is detected, the RSSR displays a sharp peak which triggered the algorithm to quickly decrease λ. Once the correct model is selected the λ returns to the maximum value, providing more accurate estimate.

3.5 Conclusions

The GNF with memory combines the minimum variance estimation and the Newton method of local linearisation to estimate the process true state. The recursive form for the Gauss-Newton filter has been derived in one compact form

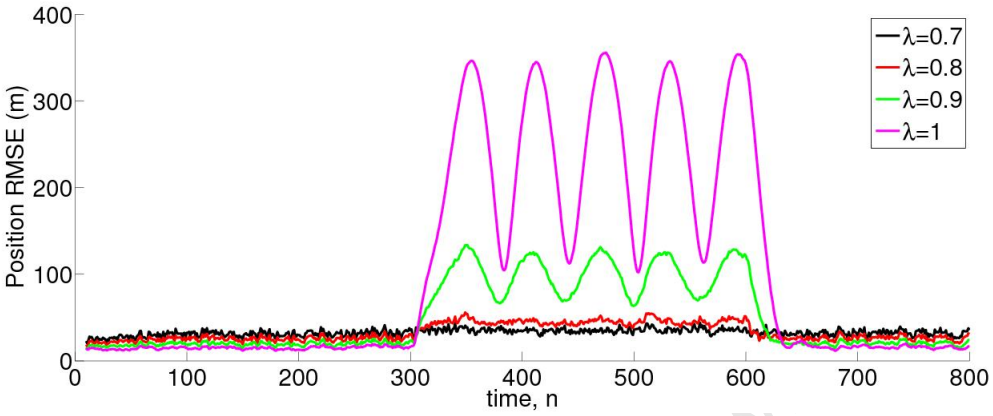

Figure 3.3: The position RMSE computed at different values of forgetting factor. Lower values of forgetting factors have higher but more stable RMSE.

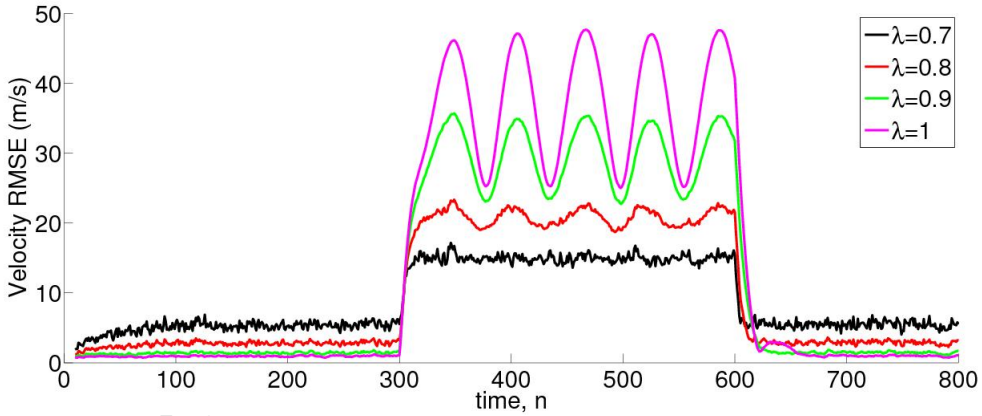

Figure 3.4: The velocity RMSE showing similar trend to Figure 3.3

that is equivalent to the iterated extended Kalman filter when the forgetting factor in the filter equation is equal to 1. We then demonstrated the importance of the forgetting factor in the stability of the recursive Gauss-Newton filter. An adaptive forgetting factor algorithm,the adaptive switching and control algorithm (ASCA), based on the recursive sum of squared residuals is developed. The ASCA allows a timely switching of the filter model to track manoeuvres. The effectiveness of the ASCA was demonstrated in a simulation study. The

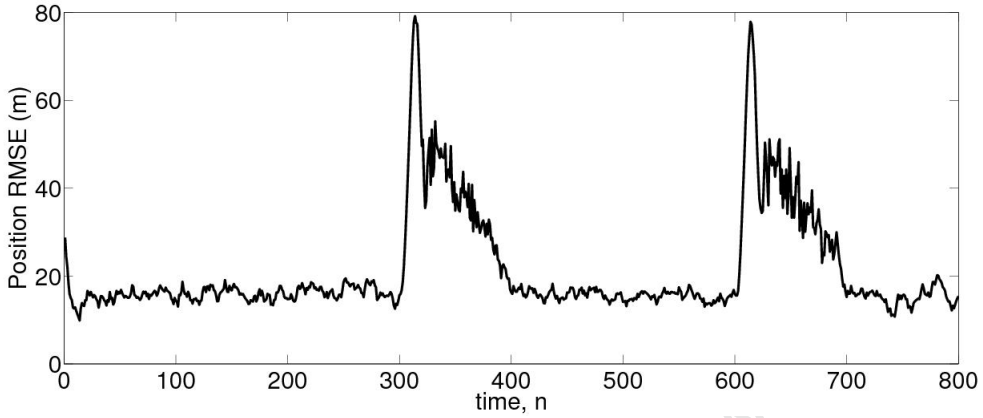

Figure 3.5: The position RMSE with ASCA. The ASCA allows the filter to switch between models without divergence

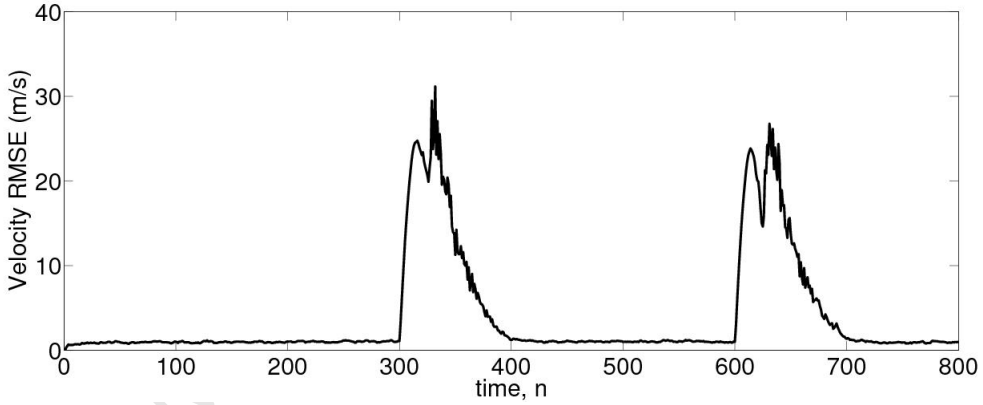

Figure 3.6: The velocity RMSE with ASCA.

forgetting factor has multiple applications which includes allowing the filter to track time varying systems and therefore the filter should include it to ensure its robustness as stated by the stability condition.

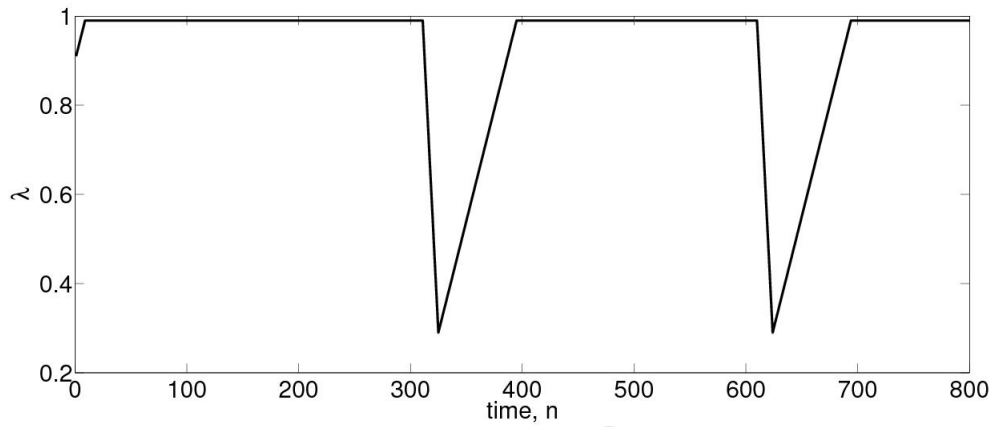

Figure 3.7: The forgetting factor sharply drops at the start of each manoeuvre allowing the filter to change model without divergence. Then it increases slowly to a maximum value for more accurate estimation

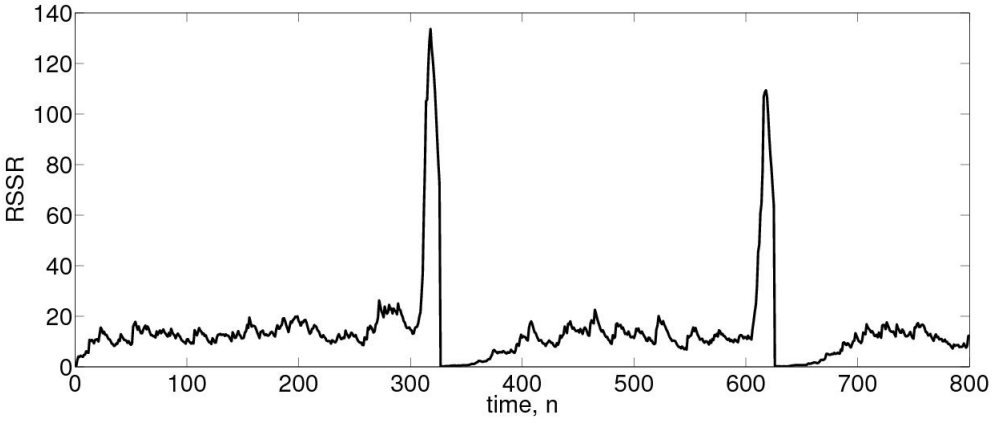

Figure 3.8: The RSSR declares manoeuvres by showing sharp peaks in values

Chapter 4

The Gauss-Newton filter and its adaptation to Levenberg Maquardt methods

This chapter shows that the Levenberg-Marquardt Algorithms (LMA) can be merged into the GN filters (recursive and non-recursive) to track difficult, non-linear trajectories, without divergence [18, 19, 20]. In the past, the LMA has been used for initialising tracking filters [21, 22, 23]. This hybrid filter is also self initialising. The LMA are optimisation techniques widely used for data fitting [12]. These optimisation techniques are iterative and guarantee convergence in a specified region i.e. they don't necessarily produce global minima [24]. They are also used in most neural network algorithm [25, 26, 27]. The GN filter is iterative and non-recursive with memory that can be adaptively controlled. In Chapter 3 we derived the recursive form of the GN filter, known as the Recursive

GN filter (RGNF) and we showed that the RGNF is equivalent to the iterated extended Kalman filter (IEKF) when unbounded filter memory length is considered. Furthermore, the stability analysis demonstrated the necessity of having finite memory length. The GN filter differs from the Gauss-Newton optimisation methods discussed in the literature as it provides a different method for computing the Hessian matrix [28]. Key to the computation of the Hessian matrix is the total transition matrix, a Jacobian matrix computed by back propagation of the current estimate over the entire memory length. This flexibility makes the GN filter filter highly suitable for tracking in strongly non-linear situations.

In this chapter we adapt the GN filter to the LMA method (which we call the Morrison LMA Filter) and we state that this filter can be used for radar target tracking with improved convergence. Similarly, we adapt the recursive GN filter to LMA method to overcome computational burden. We also show through simulation studies that the performance of both filters is not affected by the process noise whose knowledge is central to the family of Kalman filters. The literature on the use of LMA as a tracking algorithm are rare, possibly due to lack of exposure to Morrison's approach in the GN filter. The LMA is well known as an aid for track initiation [21, 22, 23]. We make it clear here that the LMA is not applied as an initiation tool in our hybrid filter, but rather as an integral part of the filter. The paper starts in Section 2.1 to define a state space model based on non-linear differential equations. Section 4.1 describes the incorporation of the LMA methods into the GN filter to produce the Morrison LMA, which converges very robustly. Section 4.2 shows a similar adaptation to the recursive GN filter. The performance of the new filter is demonstrated in a

series of simulations described in Section 4.3.

4.1 Adaptation to Levenberg Marquardt

This section represents the key step in the development of the Morrison LMA Filter. For simplicity in this adaptation of the GNF to the Levenberg-Marquard method, we assume the dynamic of the process we want to track is governed by linear differential equations and the observation scheme is non-linear. The process transition equation will be:

$$X_{n+\varsigma} = \Phi(\varsigma)X_n \tag{4.1}$$

.

The GN filter will fail to converge if the Hessian matrix is singular. By definition, this matrix is positive definite. However, it can loose this property due to numerical inaccuracy or high non-linearity. To avoid the singularity, a damping factor is introduced in Equation 2.23 as follows:

$$\delta\hat{X}_n = (\mathbf{T}_n^T\mathbf{R}_n^{-1}\mathbf{T}_n + \mu I)^{-1}\mathbf{T}_n^T\mathbf{R}_n^{-1}\delta\mathbf{Y}_n \tag{4.2}$$

which is the form suggested by Levenberg and Marquardt [12].

The effect of the damping factor is as follows:

For all positive μ the matrix $(\mathbf{T}_n^T\mathbf{R}_n^{-1}\mathbf{T}_n + \mu I)$ is positive definite, ensuring that

δX is in the descent direction;

When μ is large we have:

$$\delta\hat{X} = \frac{1}{\mu}\mathbf{T}_n^T\mathbf{R}_n^{-1}\delta\mathbf{Y}_n \tag{4.3}$$

The algorithm behaves as a steepest descent which is ideal when the current solution is far from the local minimum. The convergence will be slow but is guaranteed. When μ is small, the algorithm has faster convergence and behaves like the Gauss-Newton.

The damping factor can be updated by the gain ratio:

$$\varrho = \frac{\delta\mathbf{Y}_n^T\mathbf{R}_n^{-1}\delta\mathbf{Y}_n - (\mathbf{Y}_n - \bar{\mathbf{Y}})^T\mathbf{R}_n^{-1}(\mathbf{Y}_n - \bar{\mathbf{Y}})}{E(0) - E(\delta X_n)} \tag{4.4}$$

where $\delta\mathbf{Y}_n$ is the long vector of L sequences of observation including the current observation.

$\bar{\mathbf{Y}}$ is the long error free observation computed by back propagation of the current iterate X_{new}. If we sample at constant rate ς then:

$$\bar{\mathbf{Y}} = \begin{bmatrix} G(X_{new}) \\ G(\Phi(-\varsigma)X_{new}) \\ \vdots \\ G(\Phi(-(L-1)\varsigma)X_{new}) \end{bmatrix} \tag{4.5}$$

The numerator is the actual computed gain and the denominator is the predicted

gain. Recalling Equation 2.20 and replacing δX_n by the expression in Equation 2.23 after expansion we have:

$$E_n(0) - E_n(\delta X_n) = \delta X_n^T(\mathbf{T}_n^T\mathbf{R}_n^{-1}\delta\mathbf{Y}_n + \mu\delta X_n) \tag{4.6}$$

A large value of ϱ indicates that $E(\delta X_n)$ is a good approximation of $\bar{\mathbf{Y}}$, and μ can be decreased so that the next Levenberg-Marquardt step is closer to the Gauss-Newton step. If ϱ is small or negative then $E(\delta X_n)$ is a poor approximation, then μ should be increased to move closer to the steepest descent direction. The algorithm adapted from [29] is presented in Algorithm 3

4.2 Adaptation of the recursive GNF to Levenberg Marquardt

We consider the following definition:

$$W_n = \mathbf{T}_n^T\mathbf{R}_n^{-1}\mathbf{T}_n \tag{4.7}$$

$$\xi_n = \mathbf{T}_n^T\mathbf{R}_n^{-1}\delta\mathbf{Y} \tag{4.8}$$

In Chapter 2 we showed that these quantities can be obtained recursively as

follows:

$$W_n = \lambda A(\bar{X}_{n-1})^{-T} W_{n-1} A(\bar{X}_{n-1})^{-1} + M(\bar{X}_n)^T R^{-1} M(\bar{X}_n) \tag{4.9}$$

$$\xi_n = \lambda A(\bar{X}_{n-1})^{-T} \xi_{n-1} + M(\bar{X}_n)^T R^{-1} \delta Y_n \tag{4.10}$$

where λ is the forgetting factor and $A(\bar{X}_{n-1})$ the state sensitivity matrix.

The adaptation of the RGNF to Levenberg Marquardt method can be done similarly to the non recursive GN. We therefore replace W_n by $W_n + \mu I$. The gain ratio denominator is:

$$E_n(0) - E_n(\delta X_n) = \delta X_n^T (\xi_n + \mu \delta X_n) \tag{4.11}$$

If we define:

$$F(\delta X_n) = (Y_n - G(\bar{X}_n + \delta X_n))^T R^{-1} (Y_n - G(\bar{X}_n + \delta X_n)) \tag{4.12}$$

then the gain ratio numerator is $F(0) - F(\delta X_n)$.

In Algorithm 4 we propose an implementation of the RGNF adaptation to Levenberg-Marquardt.

4.3 Simulations

We demonstrate the efficacy of the new LMA versions of the GN filters with two simulations. The parameters of each simulation are described as part of each simulation, as well as the specific objectives.

4.3.1 The non recursive GNF

The objective of this simulation is to demonstrate the adaptation of the non recursive GNF to LMA (Algorithm 3). In the simulation studies we adopt multiple target dynamics:

Case 1 : The target is moving at constant velocity under the process noise of constant standard deviation.

Case 2 : The target is moving at constant velocity with the process noise standard deviation varying.

In all the cases, the observation scheme is non-linear. The observables are range ρ, bearing ϕ, elevation θ and Doppler f_d. The observation equation is therefore defined as follows:

$$Y = \begin{bmatrix} \sqrt{x^2+y^2+z^2} \\ tan^{-1}(y/x) \\ tan^{-1}(z/\sqrt{x^2+y^2}) \\ K_d(x\dot{x}+y\dot{y}+z\dot{z})/\sqrt{x^2+y^2+z^2} \end{bmatrix} + v(t) \tag{4.13}$$

where $v(t)$ is vector of random variables with covariance

$$R = \begin{bmatrix} 60^2 & 0 & 0 & 0 \\ 0 & 0.001^2 & 0 & 0 \\ 0 & 0 & 0.001^2 & 0 \\ 0 & 0 & 0 & 2^2 \end{bmatrix}$$

Throughout the simulations. $K_d = -2\pi/\lambda = -200$. The constants $\tau = 10^{-1}$, $\varepsilon = 10^{-20}$, $k_{max} = 200$, $\zeta = 1$ s are used in all the cases.

Case 1

In this example, we seek to demonstrate that the filter does not diverge in the presence of a constant variance process noise which is unknown to its model. The target state vector $X = [x, \dot{x}, y, \dot{y}, z, \dot{z}]^T$ is defined by the following transition equation:

$$X_{n+1} = \begin{bmatrix} 1 & \varsigma & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \varsigma & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \varsigma \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} X_n + \begin{bmatrix} \frac{1}{2}a_1\varsigma^2 \\ a_1\varsigma \\ \frac{1}{2}a_2\varsigma^2 \\ 3.a_2\varsigma \\ \frac{1}{2}a_3\varsigma^2 \\ a_3\varsigma \end{bmatrix} \tag{4.14}$$

where a_1, a_1, a_1 are independent, Gaussian random variables, with standard deviation $\sigma = 0.001$. The state vector is used in Equation **??** to generate measurements for the simulation. The filter, however, does not depend on the process

noise, it assumes the target is moving at constant speed without process noise.

The initial value of the state vector is $X = [800, 25, 1000, -25, 400, 14]$. Two thousand samples are generated and the process is repeated 50 times. The position root mean squared error (RMSE) after the 50 Monte Carlo runs is presented in Figure 4.1. The position RMSE is computed as follows :

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}((x_n^i - \hat{x_n})^2 + (y_n^i - \hat{y_n})^2 + (z_n^i - \hat{z_n})^2)} \tag{4.15}$$

where (x_n^i, y_n^i, z_n^i) and $(\hat{x_n}, \hat{y_n}, \hat{z_n})$ true and estimated position coordinates respectively.

We see from Figure 4.1 that there is no divergence in position despite the presence of the process noise, which is unknown to the filter. The filter with the smallest memory exhibits the largest RMSE. The average number of iterations is presented in Figure 4.2. All the filters have about the same value of k, which is around 34, meaning the computation time of the algorithm is primarily dependent on the computation of the $\mathbf{T}_n$ matrix.Therefore, if we want to reduce the computation time of the algorithm, we would choose a small memory length (the $\mathbf{T}_n$ matrix will be small and hence less computation required), but this would result in less accuracy in the estimates.

Case 2

Here we show the effect of higher variation in the process noise on the filter performance. In this case the target dynamic model is the same as in Case 1.

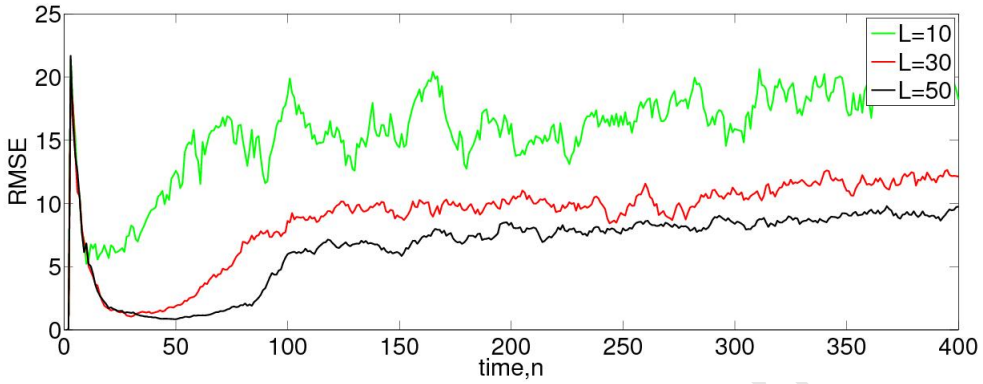

Figure 4.1: The filter with the highest memory length exhibits lowest RMSE

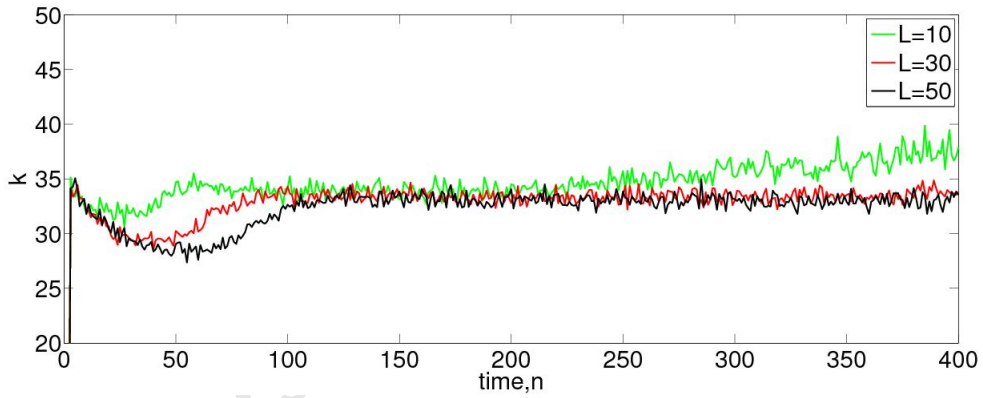

Figure 4.2: The number of iterations varies little as a function of considerable memory variation

The standard deviation(σ) of the process noise is varied. From sample 0 to 200 $\sigma = 0.001$, between samples 201 and 260 $\sigma = 0.05$ and finally from sample 261 to 400 $\sigma = 0.001$. The position RMSE after 200 Monte Carlo runs is shown in Figure 4.3. All the filters reset to the original RMSE when the process noise standard deviation returned to the former value. The RMSE of filter with the smallest memory length is less affected by these changes. However, the number of iterations during high disturbance is higher for the smaller memory length filter (Figure 4.4). Theses results highlight the adaptiveness of the algorithm to

disturbance. They also give a hint of the ability of the filter to track manoeuvres. This will be the subject of our next publication.

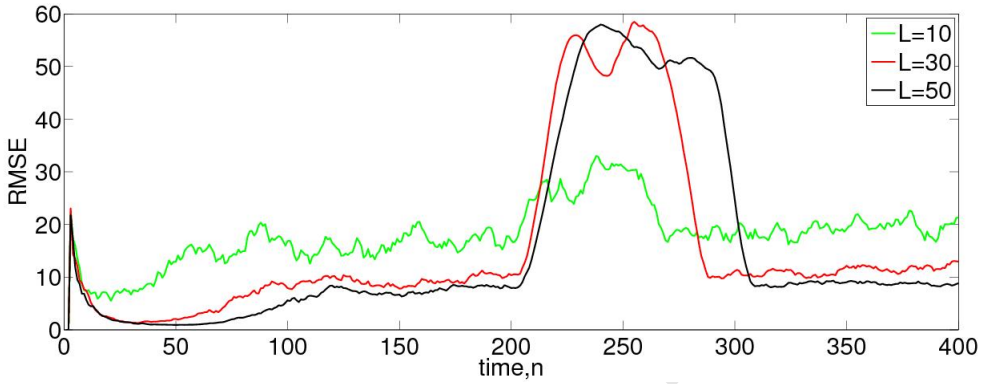

Figure 4.3: The filter with larger memory length is more sensitive to process noise variation

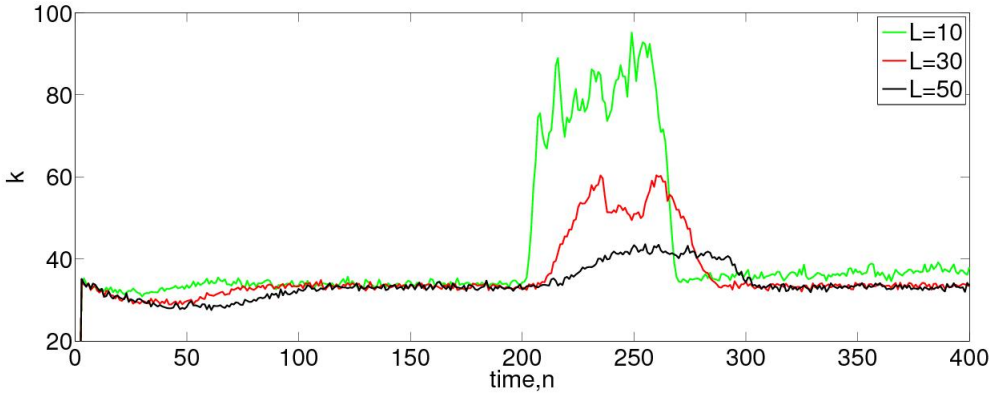

Figure 4.4: The filter with lower memory length requires higher number of iteration at higher disturbance

4.3.2 The Recursive GNF with LMA

This second simulation is to demonstrate the adaptation of the RGNF to LMA described in Algorithm 3. In these simulation studies, we consider an example

of a vehicle executing various manoeuvres. During turn manoeuvres of unknown constant turn rate, the aircraft dynamic model is:

$$X_n = \begin{bmatrix} 1 & \frac{\sin(\Omega T)}{\Omega} & 0 & -(\frac{1-\cos(\Omega T)}{\Omega}) & 0 \\ 0 & \cos(\Omega T) & 0 & -\sin(\Omega T) & 0 \\ 0 & \frac{1-\cos(\Omega T)}{\Omega} & 1 & \frac{\sin(\Omega T)}{\Omega} & 0 \\ 0 & \sin(\Omega T) & 0 & \cos(\Omega T) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} X_{n-1} + v_n \tag{4.16}$$

where the state of the vehicle is $X_n = [x, \dot{x}, y, \dot{y}, \Omega]$, with x,y the position coordinates and $\dot{x}$,$\dot{y}$ their corresponding velocity components. The process noise $v_k \sim \mathcal{N}(0, Q)$ with covariance matrix $Q = diag \begin{bmatrix} q1BB^T & q1BB^T & q2T \end{bmatrix}$ where,

$$BB^T = \begin{bmatrix} \frac{T^4}{4} & \frac{T^3}{2} \\ \frac{T^3}{2} & T^2 \end{bmatrix} \tag{4.17}$$

When the vehicle moves at a nearly constant velocity its dynamic model is:

$$X_n = \begin{bmatrix} 1 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} X_{n-1} + v_n \tag{4.18}$$

The vehicle is observed by a radar located at the origin of the plane, capable of measuring the range r and and the bearing angle θ. The measurement equation

is therefore:

$$\begin{bmatrix} r_n \\ \theta_n \end{bmatrix} = \begin{bmatrix} \sqrt{x^2+y^2} \\ tan^{-1}(\frac{y}{x}) \end{bmatrix} + w_n \tag{4.19}$$

where the measurement noise is $w_k \sim \mathcal{N}(0, R)$ with covariance $R = diag \begin{bmatrix} \sigma_r^2 & \sigma_\theta^2 \end{bmatrix}$

The following constants were used for data generation: $T = 1$ s; $\Omega = -3^\circ$ s^{-1}; $q_1 = 0.01$ m^{2}s^{-4}; $q_2 = 1.75 \times 10^{-4}$ s^{-4}; σ_r=10 m; $\sigma_\theta = \sqrt{0.1}$ mrad.

The vehicle starts at true initial state X_n=[10 m, 25 ms^{-1},400 m, 0 ms^{-1},$-3^\circ s^{-1}$] and moves at nearly constant velocity for 100 s, Then it executes a turn manoeuvre from time index $n = 101$ to $n = 150$. After the manoeuvre, the vehicle's velocity remains nearly constant from $n = 151$ to $n = 250$. At $n = 251$ it starts a new turn manoeuvre at rate $\Omega = -3^0 s^{-1}$ until $n = 400$. Finally from $n = 400$ to $n = 500$ it moves at nearly constant velocity. Figure 4.5 describes the complete trajectory of the vehicle.

The filter uses a single model of a constant velocity to track the entire manoeuvre:

$$A(X_n) = \begin{bmatrix} 1 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.20}$$

The initial value $W_{-1/0} = 10^{-2} I$, where I is an identity matrix. The filter

parameters are the following $k_{max} = 200$, $\varepsilon = 1 \times 10^{-24}$, $\tau = 1 \times 10^{-3}$, $\lambda = 0.4$

The filter initial state is generated randomly and then ensuring that it has the same sign as the true state. This procedure guarantees the local convergence of the first estimate. The experiment was repeated for 250 Monte Carlo runs and the root means squared error (RMSE) is used as a performance metric. The position RMSE is computed using the following expression:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}((x_n^i - \hat{x_n})^2 + (y_n^i - \hat{y_n})^2)} \tag{4.21}$$

where (x_n^i, y_n^i) and $(\hat{x_n}, \hat{y_n})$ true and estimated position coordinates respectively. The velocity root mean square error (RMSE) is computed similarly.

Figures 4.6and 4.7 show the RMSE of the position and velocity respectively. The position RMSE is not affected by different manoeuvres while the velocity RMSE shows variation from different manoeuvre states. The average values of the damping factor after complete cycles of iteration is presented in Figure 4.8. The damping factor increases rapidly at the transition between manoeuvres. The average number of iterations k at convergence from Figure 4.9 shows similar variations.

4.4 Conclusion

This paper introduced the standard Gauss-Newton filter that uses back propagation of the predicted state vector over a finite memory length to compute

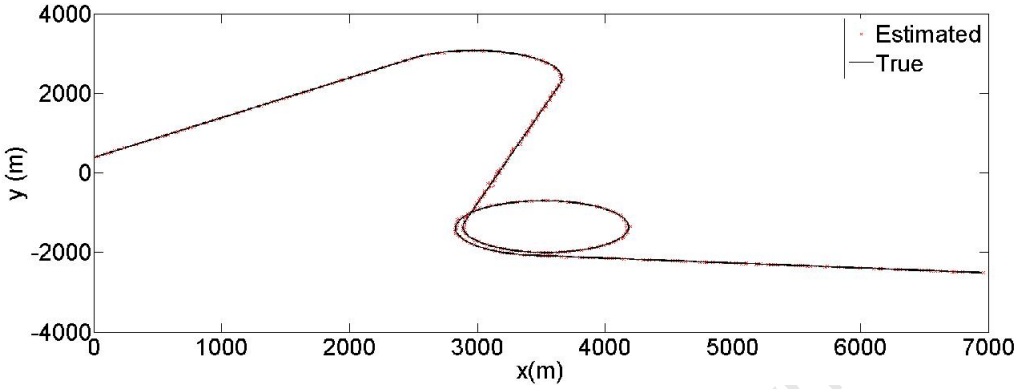

Figure 4.5: Target complete trajectory with manoeuvres.

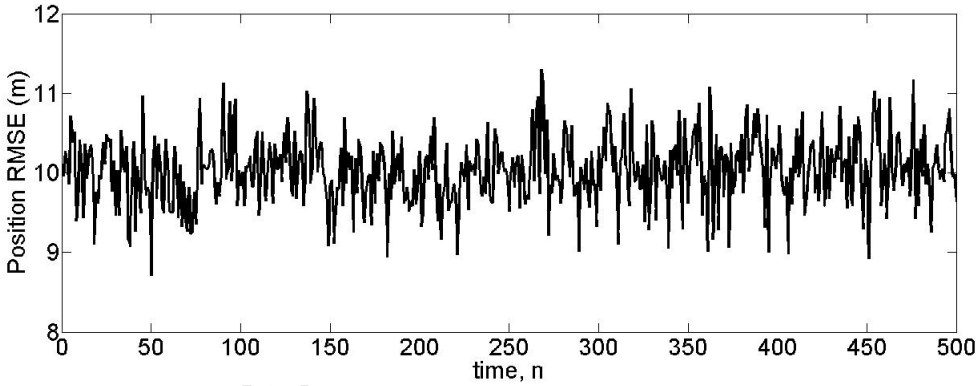

Figure 4.6: The Position RMSE is unaffected by the manoeuvres.

the Jacobian matrix. It then computes the current estimate of the state vector through minimum variance estimation. The Gauss-Newton Filter was then adapted to the Levenberg and Marquardt method to improve its convergence. Similar adaptation was applied to the recursive version of the filter.

Both GNF and RGN filters, adapted to Levenberg-Marquardt, were implemented and tested in simulation studies which showed the filters are not dependent on the process noise covariance matrix. It was also observed that both filters can use the master control algorithm described in Chapter 2 and Morrison's second

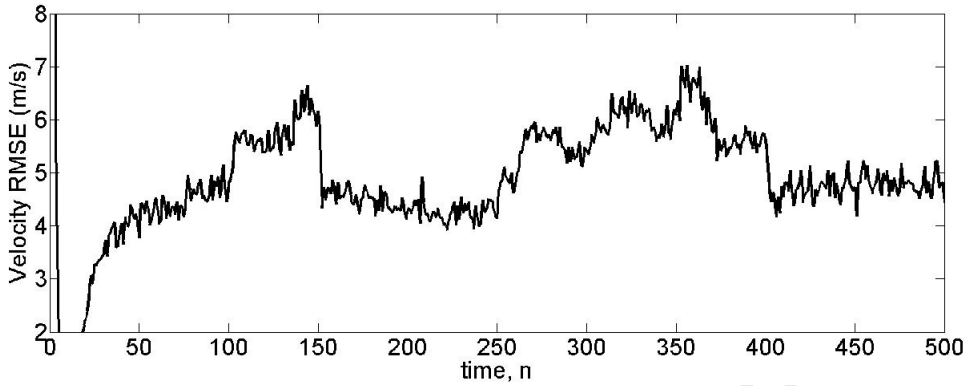

Figure 4.7: The velocity RMSE varies with manoeuvres.

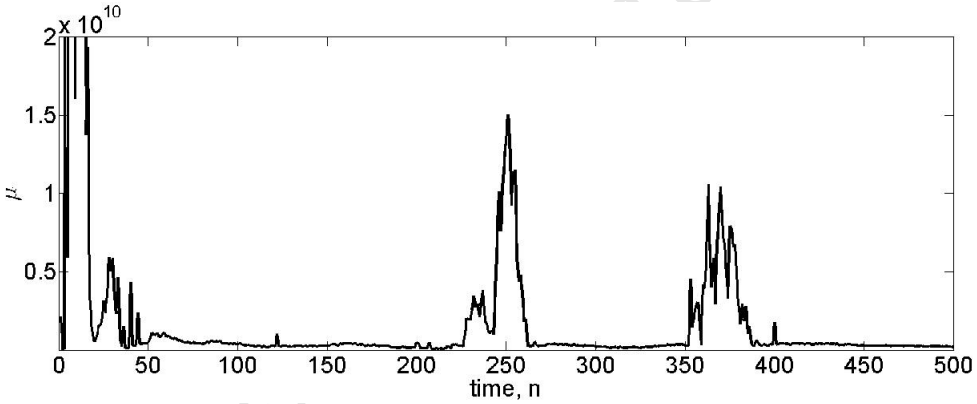

Figure 4.8: The damping factor shows sharp peaks at start of manoeuvres

book [3] to sense manoeuvres.

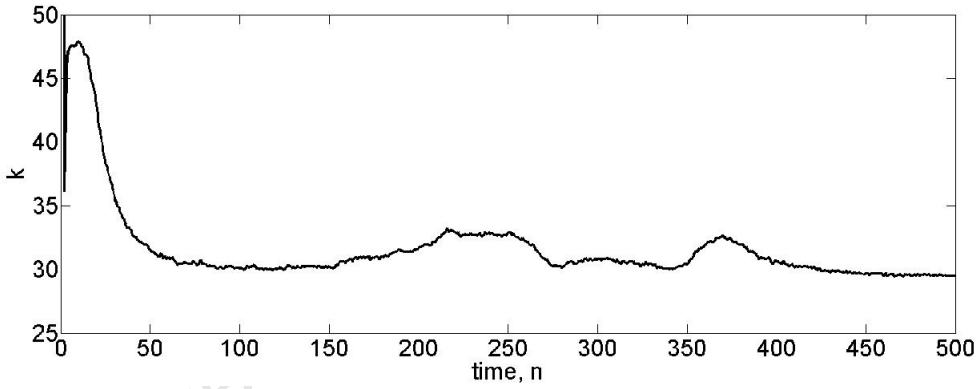

Figure 4.9: The number of iterations increases during manoeuvres.

Chapter 5

Conclusions

The GNF with memory combines the minimum variance estimation and the Newton method of local linearisation to estimate the process' true state. The recursive form for the Gauss-Newton filter has been derived in a compact form that is equivalent to the iterated extended Kalman filter (IEKF) when the forgetting factor in the filter equation is unity. Further stability analysis has shown that it is necessary to have a finite memory length, hence the RGNF and the GNF are in this context more robust filters than the IEKF 3.4. This result also explains the error covariance consistency of the GNF as opposed to the IEKF. The IEKF covariance matrix is unbounded and therefore makes the filter more susceptible to performance degradation.

An adaptive forgetting factor algorithm, the adaptive switching and control algorithm (ASCA), based on the recursive sum of squared residuals is developed. The ASCA allows a timely switching of the filter model to track manoeuvres. The effectiveness of the ASCA is demonstrated in a simulation studies.

The forgetting factor has multiple importance which includes allowing the filter to track time varying systems. The filter should therefore include it to ensure its robustness as stated by the stability condition.

The adaptation of both the GNF and the RGNF to the Levenberg-Marquardt Method (LMA) results in filters that are more robust in handling highly non-linear systems. These new filters can be used in navigation and in other areas of signal processing where the IEKF showed limitations. Both GN and RGN filters adapted to Levenberg-Marquardt were implemented and tested in simulation studies, which showed the filters are not dependent on the process noise covariance matrix. It was also observed that both filters can use the master control algorithm described to sense manoeuvres.

5.1 Future considerations

The Gauss-Newton filter remains a very robust filter with further possibilities for improvement. In Chapter 2 we showed the filter can incorporate the process noise covariance matrix which demonstrates the process noise can produce bias, providing a clearer explanation to the error covariance inconsistency seen in the Kalman filter. A thorough analysis of the effect of the Q matrix on Kalman filter can be carried out using this route.

We also showed that a biased estimator based on the linear Gauss-Newton filter can be obtained. This estimator, with optimum bias matrix, offers much lower mean squared error than the original filter. This is a novel approach to tracking filters and therefore requires detailed studies.

Data association is the process of assigning a measurement to a particular track in the presence of multiple targets or clutter. The Kalman filters and the polynomial filters[30] have been adapted to probabilistic data association (PDA) for improved performance in highly cluttered environment. Future work could be the adaptation of the RGNF to PDA with focus on the effect of the forgetting factor on the filter performance.

The Commensal Radar uses FM broadcast as transmission of opportunity to locate targets [31]. Due to their narrow band the FM signal offers very bad range resolution. However, signal processing techniques allows accurate measurement of the Doppler which is a highly non-linear function of the target state vector. Studies have shown that the GNF and the RGNF produced very good tracking performance [32, 10]. Future work can include integrating the filter algorithm into a real time operating Commensal Radar.

Appendix A

A.1 The differential equation governing $\delta X(t)$

Starting from:

$$\delta X(t) = X(t) - \bar{X}(t) \tag{A.1}$$

The differentiation rule is applied:

$$D\delta X(t) = F(\bar{X}(t) + \delta X(t)) - F(\bar{X}(t)) \tag{A.2}$$

Let F be defined as follows :

$$F = \begin{bmatrix} f_1 \\ . \\ . \\ . \\ f_n \end{bmatrix} \tag{A.3}$$

Equation becomes:

$$D\delta X(t) = \begin{bmatrix} f_1(\bar{X}(t) + \delta X(t)) \\ . \\ . \\ . \\ f_n(\bar{X}(t) + \delta X(t)) \end{bmatrix} - \begin{bmatrix} f_1(\bar{X}(t)) \\ . \\ . \\ . \\ f_n(\bar{X}(t)) \end{bmatrix} \tag{A.4}$$

The Taylor first order approximation is applied:

$$D\delta X(t) = \begin{bmatrix} f_1(\bar{X}(t)) \\ . \\ . \\ . \\ f_n(\bar{X}(t)) \end{bmatrix} + \begin{bmatrix} \nabla f_1(\bar{X}(t))^T \\ . \\ . \\ . \\ \nabla f_n(\bar{X}(t))^T \end{bmatrix} \delta X(t)$$
$$- \begin{bmatrix} f_1(\bar{X}(t)) \\ . \\ . \\ . \\ f_n(\bar{X}(t)) \end{bmatrix} \tag{A.5}$$

The following relation is obtained :

$$D\delta X(t) = A(\bar{X}(t))\delta X(t) \tag{A.6}$$

Where:

$$A(\bar{X}(t)) = \begin{bmatrix} \nabla f_1(\bar{X}(t))^T \\ . \\ . \\ . \\ \nabla f_n(\bar{X}(t))^T \end{bmatrix} = \frac{\partial F(X(t))}{\partial (X(t))}\Big|_{\bar{X}(t)} \tag{A.7}$$

A.2 The relation between δX_n and δY_n

$$\delta Y_n = G(\bar{X}_n + \delta X_n) - G(\bar{X}_n) \tag{A.8}$$

As direct consequence of A.1 the following relationship is obtained:

$$\delta Y_n = M(\bar{X}_n)\delta X_n + v_n \tag{A.9}$$

A.3 Positive definite matrices inequality

According to Berstein matrix mathematics [14], for positive definite and hermitian matrices A and B the following is true:

- $A < B$ then $trace(A) < trace(B)$

- $A < B$ then $B^{-1} < A^{-1}$

Using the above facts we have :

- $(R+M^TQM) > R$ leading to $(R+M^TQM)^{-1} < R^{-1}$ or $(R+M^TQM)^{-1}R < R^{-1}R$
- $trace((R + M^TQM)^{-1}R) < trace(R^{-1}R)$
- $(R + M^TQM)^{-1} < R^{-1}$ leads $\mathbf{R}(\mathbf{Q})^{-1} < \mathbf{R}^{-1}$

A.4 The recursive filter

Algorithm 1 Basic iterative form

$k := 1; \bar{X}_n := X_{n/n-1};$

$\delta Y_n := Y_n - G(\bar{X}_n);$

$\mathbf{W}_n = \mathbf{W}_{n-1/n} + M(\bar{X}_n)^T R^{-1} M(\bar{X}_n)$

$\xi_n = \xi_{n/n-1} + M(\bar{X}_n)^T R^{-1}(Y_n - G(\bar{X}_n));$

$\delta\hat{X}_n = \mathbf{W}_n^{-1}\xi_n;$

$X_n = \bar{X}_n + \delta\hat{X}_n;$

$\bar{X}_n = X_n;$

While $(k \leq k_{max})$

$k := k + 1;$

$\mathbf{W}_n = \mathbf{W}_{n/n-1} + M(\bar{X}_n)^T R^{-1} M(\bar{X}_n)$

$\xi_n = \mathbf{W}_{n/n-1}(\hat{X}_{n/n-1} - \bar{X}_n) + M(\bar{X}_n)^T R^{-1}(Y_n - G(\bar{X}_n));$

$\delta\hat{X}_n = \mathbf{W}_n^{-1}\xi_n;$

$X_n = \bar{X}_n + \delta\hat{X}_n;$

$\bar{X}_n = X_n;$

endwhile

$X_{n+1/n} = F(X_n);$

$\mathbf{W}_{n+1/n} = \lambda A(X_{n/n})^{-T}\mathbf{W}_n A(X_{n/n})^{-1};$

$\xi_{n+1/n} = \lambda A(X_{n/n})^{-T}\xi_n;$

Algorithm 2 IEKF when $\lambda = 1$

$k := 0; \bar{X}_n := X_{n/n-1};$

While $(k \leq k_{max})$

$k := k + 1;$

$K_n = W^{-1}_{n/n-1} M(\bar{X}_n)^T [R + M(\bar{X}_n) W^{-1}_{n/n-1} M(\bar{X}_n)^T]^{-1}$

$\delta \hat{X}_n = K_n [Y_n - G(\bar{X}_n) - M(\bar{X}_n)(\hat{X}_{n/n-1} - \bar{X}_n)];$

$X_n := \bar{X}_n + \delta \hat{X}_n;$

$\bar{X}_n = X_n;$

endwhile

$W^{-1}_n = \lambda^{-1} [I - K_n M(\bar{X}_n)] W^{-1}_{n/n-1};$

$X_{n+1/n} = F(X_n);$

$W_{n+1/n} = \lambda^{-1} A(X_{n/n}) W^{-1}_n A(X_{n/n})^T + Q;$

A.5 The adaptation to Levenberg-Marquardt

Algorithm 3 Levenberg=Marquardt algorithm for the non recursive GN Filter.

$k := 0, \nu := 2, X := X_{n/n-1}$
$A := \mathbf{T}_n^T \mathbf{R}_n^{-1} \mathbf{T}_n; \delta \mathbf{Y}_n := \mathbf{Y}_n - \bar{\mathbf{Y}}_n$; $g := \mathbf{T}_n^T \mathbf{R}_n^{-1} \delta \mathbf{Y}_n$; $\bar{\mathbf{Y}}_n$ is computed using X
$stop := false; \mu = \tau \times max(diag(A))$;
While (not stop) and $(k \leq k_{max})$
  $k := k + 1$
  repeat
  solve $(A + \mu I)\delta \hat{X} 3._n = g$
  if $(||\delta \hat{X}_n|| \leq \varepsilon ||X||)$
    stop:=true;
  else
    $X_{new} := X + \delta \hat{X}_n$;
      $\varrho = [\delta \mathbf{Y}_n^T \mathbf{R}_n^{-1} \delta \mathbf{Y}_n - (\mathbf{Y}_n - \bar{\mathbf{Y}})^T \mathbf{R}_n^{-1} (\mathbf{Y}_n - \bar{\mathbf{Y}})] / [\delta \hat{X}_n^T (g + \mu \delta \hat{X}_n)]$; $\bar{\mathbf{Y}}$ evaluated at X_{new}
      if $\varrho > 0$
        $X = X_{new}$;
          $A := \mathbf{T}_n^T \mathbf{R}_n^{-1} \mathbf{T}_n; \delta \mathbf{Y}_n := \mathbf{Y}_n - \bar{\mathbf{Y}}_n$; $g := \mathbf{T}_n^T \mathbf{R}_n^{-1} \delta \mathbf{Y}_n$; $\bar{\mathbf{Y}}_n$ is computed using X
          $\mu = \mu \times max(1/3, 1 - (2\varrho + 1)^3); \nu := 2$;
      else
        $\mu := \nu \times \mu$;
        $\nu := 2 \times \nu$;
      endif
    endif
  until$(\varrho > 0)$or(stop);
endwhile
$X_{n/n} = X$;
$X_{n+1/n} = F(X_{n/n})$

Algorithm 4 Levenberg-Marquardt applied into the Recursive GN.

$k := 0; \nu := 2; \bar{X}_n := X_{n/n-1};$
$\delta Y_n = Y_n - G(\bar{X}_n);$
$\mathbf{W}_n = \mathbf{W}_{n/n-1} + M(\bar{X}_n)^T R^{-1} M(\bar{X}_n)$
$\xi_n = \xi_{n/n-1} + M(\bar{X}_n)^T R^{-1} \delta Y_n;$
$stop := false; \mu = \tau \times max(diag(\mathbf{W}_{n/n-1}));$
While (not stop) and $(k \leq k_{max})$
  $k := k + 1;$
  repeat;
  solve $(\mathbf{W}_n + \mu I)\delta\hat{X}_n = \xi_n;$
  if $(||\delta\hat{X}_n|| \leq \varepsilon||\bar{X}_n||)$
    stop:=true;
  else
    $X_{new} := \bar{X}_n + \delta\hat{X}_n;$
    $F(\delta X) = Y_n - G(X_{new}); F(0) = \delta Y_n^T R^{-1} \delta Y_n;$
    $E(0) - E(\delta X_n) = \delta X_n^T(\xi_n + \mu\delta X_n);$
    $\varrho = \frac{F(0)-F(\delta X_n)}{E(0)-E(\delta X)};$
    if $\varrho > 0$
      $\bar{X}_n = X_{new};$
      $\delta Y_n := Y_n - G(\bar{X}_n);$
      $\mathbf{W}_n = \mathbf{W}_{n/n-1} + M(\bar{X}_n)^T R^{-1} M(\bar{X}_n);$
      $\xi_n = \mathbf{W}_{n/n-1}(\hat{X}_{n/n-1} - \bar{X}_n) + M(\bar{X}_n)^T R^{-1} \delta Y_n;$
      $\mu = \mu \times max(1/3, 1 - (2\varrho + 1)^3); \nu := 2;$
    else
      $\mu := \nu \times \mu;$
      $\nu := 2 \times \nu;$
    endif
  endif
  until$(\varrho > 0)$or(stop);
endwhile
$X_{n/n} = X_{new};$
$X_{n+1/n} = F(X_{n/n});$
$\mathbf{W}_{n+1/n} = \lambda A(X_{n/n})^{-T} \mathbf{W}_n A(X_{n/n})^{-1};$
$\xi_{n+1/n} = \lambda A(X_{n/n})^{-T} \xi_n;$

Bibliography

[1] J. Valappil and C. Georgakis, "A systematic tuning approach for the use of extended kalman filters in batch processes," in *American Control Conference, 1999. Proceedings of the 1999*, vol. 2, pp. 1143 –1147 vol.2, jun 1999.

[2] T. Lefebvre, H. Bruyninckx, and J. D. Schutter, "Kalman filters for non-linear systems: a comparison of performance," *International Journal of Control*, vol. 77, no. 7, pp. 639–653, 2004.

[3] N. Morrison, *Tracking Filter Engineering The Gauss-Newton and Polynomial Filters*. Stevenage SG1 2SD,UK: IET, 2012.

[4] N. Morrison, *Introduction to sequential smoothing and prediction*. McGraw-Hill Book Company, 1969.

[5] J.-P. da Conceicao, "Accelerating Gauss-Newton filters on FPGAs," Master's thesis, University of Cape Town, Dec. 2011.

[6] J. Milburn, "Co-processor offloading applied to passive coherent location with doppler and bearing data," master's thesis, University of Cape Town -

RRSG, February 2010.

[7] R. Nadjiasngar and M. Inggs, "Gauss-Newton filtering incorporating Levenberg-Marquardt methods for tracking ," *Digital Signal Processing*, no. 0, pp. –, 2013.

[8] Y. Zhu and X. Li, "Recursive least squares with linear constraints," in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, vol. 3, pp. 2414 –2419 vol.3, 1999.

[9] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*. New Jersey, USA: Prentice Hall, second ed., 2000.

[10] N. Morrison, R. T. Lord, and M. R. Inggs, "The Gauss-Newton algorithm applied to track-while-scan radar," in *Proceedings of the IET International Conference on Radar Systems (RADAR 2007)*, Institution for Engineering and Technology, October 2007.

[11] S. Kay and Y. Eldar, "Rethinking biased estimation [lecture notes]," *Signal Processing Magazine, IEEE*, vol. 25, no. 3, pp. 133–136, 2008.

[12] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. pp. 431–441, 1963.

[13] M. B. Malik, "State-space recursive least-squares: Part i," *Signal Processing*, vol. 84, pp. 1709–1718, 2004.

[14] D. S. Bernstein, *Matrix Mathematics Theory, Facts, and Formulas*. Oxford, UK: Princeton University Press, 2009.

[15] R. Shorten and K. Narendra, "On common quadratic Lyapunov functions for pairs of stable LTI systems whose system matrices are in companion form," *Automatic Control, IEEE Transactions on*, vol. 48, pp. 618 – 621, april 2003.

[16] Mohammed Dahleh, Munther A. Dahleh, George Verghese, "Internal stability for LTI."

[17] S. Akhtar and D. S. Bernstein, "Lyapunov-stable discrete-time model reference adaptive control," *International Journal of Adaptive Control and Signal Processing*, vol. 19, no. 10, pp. 745–767, 2005.

[18] L. Perea, J. How, L. Breger, and P. Elosegui, "Nonlinearity in sensor fusion: Divergence issues in EKF, modified truncated SOF, and UKF," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 6514, 20 - 23 August 2007 2007.

[19] R. Niu, P. Varshney, M. Alford, A. Bubalo, E. Jones, and M. Scalzo, "Curvature nonlinearity measure and filter divergence detector for nonlinear tracking problems," in *Information Fusion, 2008 11th International Conference on*, pp. 1 –8, 30 2008-july 3 2008.

[20] X. Wang and Y. Huang, "Convergence Study in Extended Kalman Filter-Based Training of Recurrent Neural Networks," *Neural Networks, IEEE Transactions on*, vol. 22, pp. 588 –600, april 2011.

[21] L. Zong-xiang and X. Wei-xin, "A new method for track initiation in a distributed passive sensor network," in *Signal Processing, 2008. ICSP 2008. 9th International Conference on*, pp. 2616 –2619, oct. 2008.

[22] M. Yeddanapudi, Y. Bar-Shalom, K. Pattipati, and S. Deb, "Ballistic missile track initiation from satellite observations," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 31, pp. 1054 –1071, jul 1995.

[23] P. Howland, "Target tracking using television-based bistatic radar," *Radar, Sonar and Navigation, IEE Proceedings* -, vol. 146, pp. 166 –174, jun 1999.

[24] C. Ma and L. Jiang, "Some research on Levenberg-Marquardt method for the nonlinear equations," *Applied Mathematics and Computation*, vol. 184, no. 2, pp. 1032 – 1040, 2007.

[25] B. G. Kermani, S. S. Schiffman, and H. T. Nagle, "Performance of the Levenberg-Marquardt neural network training method in electronic nose applications," *Sensors and Actuators B: Chemical*, vol. 110, no. 1, pp. 13 – 22, 2005.

[26] E. Derya and beyli, "Analysis of EEG signals by implementing eigenvector methods/recurrent neural networks," *Digital Signal Processing*, vol. 19, no. 1, pp. 134 – 143, 2009.

[27] V. Singh, I. Gupta, and H. Gupta, "ANN-based estimator for distillation using Levenberg-Marquardt approach," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 2, pp. 249 – 259, 2007.

[28] T. Dahlin and M. Loke, "Resolution of 2D Wenner resistivity imaging as assessed by numerical modelling," *Journal of Applied Geophysics*, vol. 38, no. 4, pp. 237 – 249, 1998.

[29] O. T. K. Madsen, H.B. Nielsen, *Method of non-linear least squares problems.* Technical University of Denmark: Informatics and Mathematical Modelling, 2 ed., 2004.

[30] R. Nadjiasngar, M. Inggs, Y. Paichard, and N. Morrison, "A new probabilistic data association filter based on composite expanding and fading memory polynomial filters," in *Radar Conference (RADAR), 2011 IEEE*, pp. 152 –156, may 2011.

[31] M. Inggs and C. Tong, "Commensal radar using separated reference and surveillance channel configuration," *Electronics Letters*, vol. 48, no. 18, pp. 1158–1160, 2012.

[32] R. Nadjiasngar, S. Middleton, and M. Inggs, "Doppler-only tracking with the recursive Gauss-Newton filter," in *Radar Systems (Radar 2012), IET International Conference on*, pp. 1–5, 2012.